图灵交互设计丛书

Microinteractions

Designing with Details



微交互

细节设计成就卓越产品



[美] *Dan Saffer* 著

Don Norman 序

李松峰 译

O'REILLY®

人民邮电出版社
POSTS & TELECOM PRESS

数字版权声明

图灵交互设计丛书

微交互：
细节设计成就卓越产品

Microinteractions: Designing with Details

[美] Dan Saffer 著
Don Norman 序
李松峰 译

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo
O'Reilly Media, Inc.授权人民邮电出版社出版

人民邮电出版社
北京

图书在版编目（C I P）数据

微交互 ：细节设计成就卓越产品 / (美) 塞弗
(Saffer,D.) 著 ；李松峰译. -- 北京 ：人民邮电出版
社，2013.10
（图灵交互设计丛书）
书名原文：Microinteractions:designing with
details
ISBN 978-7-115-33161-8

Ⅰ. ①微… Ⅱ. ①塞… ②李… Ⅲ. ①电子产品－产
品设计 Ⅳ. ①TN602

中国版本图书馆CIP数据核字(2013)第223518号

内 容 提 要

一个深受用户喜爱的好产品和一个让人觉得凑合用的一般产品，区别往往就在于微交互。本书要讨论一种不同的思维方式：撇开壮观宏大、由上而下的设计，重视自底向上、精心构思的细节。这些小地方是设计师最擅长的地方，举手之劳，就能立杆见影。这是改变世界的另一种方式：在看似无关紧要的时刻释放令人心动的体验！全书共 6 章和一个附录，第 1 章介绍了微交互的概念、历史和结构，第 2 章到第 5 章剖析了构成微交互的几个要素：触发器、规则、反馈、循环与模式。第 6 章通过 3 个实例将微交互设计的所有要点融会贯通，付诸实践。附录探讨了微交互的测试问题。

本书适合产品经理、设计师、程序员、研究人员、评论人员和公司老总阅读，它能引发你的思考，提高你的品味，增强你（还有你们公司）的竞争力。

◆ 著　[美] Dan Saffer
译　李松峰
责任编辑　毛倩倩
执行编辑　张　庆
责任印制　焦志炜

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　印刷

◆ 开本：880×1230　1/32
印张：5.75
字数：175千字　2013年 10 月第 1 版
印数：1－4 000册　2013年 10 月北京第 1次印刷
著作权合同登记号　图字：01-2013-6740号

定价：29.00元

读者服务热线：(010)51095186转604　印装质量热线：(010)67129223

反盗版热线：(010)67171154

广告经营许可证：京崇工商广字第 0021 号

版权声明

O’Reilly Media, Inc.介绍

O’Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O’Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O’Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O’Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 Make 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O’Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O’Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项 O’Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O’Reilly Radar 博客有口皆碑。”

——Wired

“O’Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O’Reilly Conference 是聚集关键思想领袖的绝对典范。”

——CRN

“一本 O’Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去，Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

对本书的赞誉

“《微交互》是我期待已久的一本书。我一直都希望看到一本能够在原子级别上，深刻而又简洁地把交互设计讲清楚的书。这本书就是！”

——Jared Spool

User Interface Engineering（uie.com）

“当今世界，人们每天都要接触上百种不同的用户体验，只有最明确的交互才能让人过目不忘。《微交互》是创造这种用户体验的必读佳作。

“在这本书里，Dan Saffer 把‘认知走查’（Cognitive Walkthrough）提升到了更高层次，创建了一个新的交互模型，并示范了瞬间之美是怎么产生的。

“没有难懂的词汇，让人一目了然，真是一本难得的好书。《微交互》揭示了重要的设计原理，并通过实际的例子说明了该做什么和不该做什么。哪怕只看两页，也会受益匪浅，如果多看一遍，收获更大。这本书本身就是微交互的精彩范例。通俗易懂，平易近人，深刻的洞见总能不期而遇。”

——Kevin Fox

Gmail 设计师

“Saffer 这本书写得太精彩了，言简意赅！它讨论了一个被设计行业长期忽视的主题：在人类与技术每天的交互中，如何创建短暂、微小却

令人过目难忘的瞬间。我向所有设计师和程序员推荐这本书，因为你们肯定希望自己设计和开发的应用、网站或者其他产品，既能够让用户在使用时妙趣横生，又能帮他们解决实际问题。这本书会为你指明改进的方向，每次一个微交互。”

——Robert Reimann

Interaction Design Association (IxDA) 创始人兼总裁

PatientsLikeMe 首席交互设计师

About Face 3（Wiley）[1] 合著者

“享有魔术师中的魔术师美誉的 Dariel Fitzkee 说过‘魔术成败一看细节，二看表演’。交互设计也是这样，正是它在用户头脑中创造出了微小、妙不可言而又有意思的假象。Dan 这本《微交互》就像一个放大镜，它把这些时刻放大，而且梳理出了一套从优秀到卓越的规则。我向在任何媒体中负责设计和实现用户体验的朋友强烈推荐这本书。”

——Bill Scott

Paypal 资深 UIE 总监

《Web 界面设计》合著者

“我从来没有看到过其他任何一本书是在这个层面上剖析交互设计的。这是我读过的非常好的一本讲交互设计的书。我会多买一些送给我的设计师和产品经理，让他们看完之后再讲给我听。”

——Christian Crumlish

CloudOn 产品总监

“Dan Saffer 这本《微交互》是我几年来看过的最好的一本讲设计的书。这本书本身也是一个精彩的微交互作品，它短小精悍、构思缜密、言简意赅、妙语连珠。很多讲设计的书都只是泛泛而谈，流于抽象和自我。Saffer 却抓住了设计界绝对必须解决的要害问题，他写了一本每一个设计师都必须看的书。”

——Scott Berkun

Internet Explorer 1.0~5.0 开发者

著名畅销书作者、演讲家

注 1：中文版《About Face3 交互设计精髓》由电子工业出版社出版。

目录

唐·诺曼推荐序

一次在巴西举办的会议上，我知道了 Dan Saffer 正在研究微交互（microinteraction）。这个新词儿一下子把我给吸引住了。Dan 开场先讲了一个在交响音乐会上手机闹铃的故事。这个故事让我意识到，关注细节，就是关注全局。

接下来，使用苹果的图片分类及编辑应用 Aperture 的挫折经历，再次让我体会到把细节做好的重要性。当时，我正把一本书的所有插图往一块儿弄，突然电脑死机了。没办法，只能重新启动。重启之后再打开 Aperture，结果看到了一个声明，说数据已经损坏，然后程序就自动关闭了。啊？给出一条错误消息就算完事啦，为什么不告诉人家采取什么措施补救？我该怎么办？

我搜索 Aperture 的帮助文件，却一无所获。又搜索苹果网站，还是一无所获。我最担心也最关心的问题是：*怎么把图片找回来？*可现实是程序已经无法启动了。另一台电脑上是有备份，可我的同步程序效率太高了，被破坏的文件已经被同步到了那台电脑上。

最后，历尽艰辛，我总算在网上找到一个解决方案。这个方案的来源也是苹果网站，而且格式清晰，指令明确。按照这个说明，我只用了 15 分钟就把所有图片都恢复了。（注意，我在苹果官网上没找到这个方案，是有人在某讨论组中贴过这么一个指向苹果网站的链接。）

为什么要提这件事呢？因为如果苹果的开发人员看过这本书，我就不会那么痛苦了。微交互啊！就是把细节做好。

为什么那条错误消息里不给出解决方案及对问题的描述？毕竟，苹果对这个问题是有非常详尽的解决方案的。假如那条错误消息说："数据库损坏了。要解决此问题，请按以下步骤来做。"然后在消息框中给出一个按钮，引导用户开始按步操作……苹果为什么没这么做？难道负责开发这个功能的程序员认为这事不归他管，还是说这些程序员是维护数据库的，所以只知道出了问题不知道如何解决？再或者因为编写错误提示的人压根儿就不知道还应该给出解决方案？（我个人认为这三种可能性都有。）不管什么原因，结果都是用户体验不好。这个挫折至今都让我对 Aperture 敬而远之，宁可找其他代用品。苹果不可能希望自己的客户都这样。如果他们能看看这本书就好了。

微交互就是细节吗？一点没错，奥秘全在细节中。

"微交互"中的"微"表明要关注细节。细节？没错。不重要？错！《微交互》关注的是那些至关重要的细节，它决定了用户体验是友好的，还是令人皱眉的。正如 Dan Saffer 在前言中指出的，设计师倾向于把握大的方向。这种感觉非常好。没有什么问题大到把握不了。然而，就算大的方向没错，如果细节处理不当，那结果还是失败。别忘了，细节决定了每时每刻的体验！正是这毫厘之间的细节，成就了整个产品流畅的交互体验。反过来说，恰恰是对这些细节的忽视，导致了用户的挫折、愤怒，直至对产品产生强烈的厌恶。没错，大方向、全局观是重要，但小方向、细节观同样不容忽视。是否关注细节决定产品的成败。

实现优秀的微交互有几个步骤。首先，对有些程序员来说，就是要搞清状况。为此，必须要有极强的观察能力，看看别人如何互动，再看看自己的交互习惯，找到突破口，从逻辑上分清先后，再决定怎么做最恰当。观察什么？比如错误提示，比如对话框，它们都会呈现出一些信息，而这些信息通常会暗示下一步可以做什么。那么，为什么不在呈现信息这一步就包含"下一步"按钮呢？

要想设计出优秀的微交互，就要了解产品的最终用户，知道他们想达成什么目的，需要经历哪些步骤。还要求理解不同情形下交互的操作环境。所以，对用户的移情或者同理心，观察用户的本事以及把产品的方方面面（或者把不同编程团队甚至不同部门的产出）融入交互细节的能力都至关重要。这本书第 1 章非常清楚地介绍了做到这一切的原则。遍及全书各处的例子也折射出了微交互承载的机会。看完本书，你会发现一切还是取决于你自己。每个人自己的观察体悟是发现新的微交互机会的源泉。最关键的是，假如方案要求跨公司跨部门协作，千万不要像苹果的开发人员那样各自为战。记住，为用户考虑周详才是伟大产品的真义所在。

让微交互赢得用户的第二步就是实现。这一步需要考虑诸多设计问题：触发、规则、反馈、循环，还有模式。这些本书都一一给出了精彩的诠释。

微交互真有那么重要吗？好，我再给你讲一讲我最近花了不少钱买的一辆新车。每次我走到车前，握住门把手，把手上的灯就会亮起，车内的灯也会打开。门开了，我坐进车里，座椅、后视镜会自动调到让我感觉舒适的位置，甚至收音机都调到了我喜欢的频道。打开门，我座位上的灯就会点亮。我的乘客打开门，相应座位上的灯也会点亮。要是我和妻子轮流开车，每次换人，车都会根据我们各自的偏好重新设置。设计师是怎么设计出这一系列自动控制功能的？他们怎么知道该控制什么，不该控制什么？凭借他们的智慧，这才有了智能的微设计。这些都是细节吗？是。我们可以手工完成这些设置吗？可以。但是，当车子主动为我们做了这些事之后，坐到车里你会有一种情不自禁的喜悦，一种因为拥有这样的车而产生的自豪感。这难道不是所有汽车制造商都希望自己的客户能体验到的感觉吗？

为细节欢呼吧，要知道我们每天都生活在细节当中。我们不必再为下一步该做什么而浪费分分秒秒，为此喝彩吧，跟那些粗陋的过渡设计说再见。为 Dan Saffer 和他的这本书叫好吧，通俗易懂的讲解再配上丰富的示例，简直无敌了！我自认为在观察人们与技术产品交互方面颇有心得，但读了这本书之后，突然发现自己的能力又增长了一大

截。现在，我更加关注细节，更加关注被忽视的机会。我也知道了什么产品的哪些细节做得好。知道好与不好，是做好产品的第一步。

现在轮到你了。拿出行动来，去征服世界吧！让我们的生活更简单，更丰富多彩。让微交互从理论付诸实践。

唐·诺曼简介

唐·诺曼（Donald Arthur Norman），美国认知心理学家、美国计算机协会会士、工业设计家，认知科学学会的发起人之一。现为美国西北大学计算机科学系和心理学教授，是尼尔森 – 诺曼集团（Nielsen Norman Group）咨询公司的创办人之一，曾担任苹果公司先进技术部副总裁。1999 年，Upside 杂志提名诺曼博士为世界 100 精英之一。2002 年，诺曼获得了人机交互专家协会（SIGCHI）授予的终身成就奖。2006 年，他获得了计算机和认知科学“富兰克林奖章”。

前言

本书主题

开启家电、登录网站、在手机应用里查询天气，这些都是微交互的例子。微交互只涉及一种使用场景，只做一件事。这种功能本身就可以做成一个应用，也可以只是某应用众多功能中的一个。真正优秀的微交互高效、顺畅、时尚，而且处处体现对用户需求和意图的心领神会。一个深受喜爱的好产品和一个让人觉得凑合用的一般产品，区别往往就在于微交互带给人的体验。本书的目的就是深入分析微交互，帮助读者设计出优秀的产品。我们首先会介绍微交互的模型，然后逐章详细讨论该模型的一个方面，力求揭示出每一种微交互最优化背后的原理。如此一来，读者就可以知道从哪些地方着手改进自己的产品，从而增加用户黏性，塑造客户（及品牌）忠诚度。

读者对象

本书写给那些希望把产品——特别是数字化产品，做得更出色的人。产品经理、设计师、程序员、研究人员、评论家和公司老总都是本书的读者对象，本书能引发你的思考，提高你的品味，增强你的竞争力。

如果你认为某个小功能值得实现，它能让产品更加出色，并为此想说

服自己的客户、开发人员、产品或项目经理，那本书尤其适合你！现在，这个小功能有了一个名字：微交互。你可以更加胸有成竹地据理力争了。

本书内容

这是一本小书，讨论因小见大的小功能。

- 第 1 章　设计微交互
  介绍微交互的概念，解释为什么某些不起眼的地方那么重要。这一章会讨论微交互的构成，展示微交互要遵循的总原则。最后还会讨论微交互与项目集成的方式方法。

- 第 2 章　触发器
  介绍触发器，也就是发起微交互的装置。主要讨论手工触发（用户启动）和系统触发，以及提前展示数据等规则。

- 第 3 章　规则
  讨论影响微交互的不可见因素，以及这些规则的由来、贯彻方法，包括不要从零开始。

- 第 4 章　反馈
  讨论正反馈，也就是用户怎么理解潜规则。有三种反馈形式：视觉反馈、听觉反馈和触觉反馈。介绍以人为本和使用被人忽视的东西作为传达信息的手段等原理。

- 第 5 章　循环与模式
  讨论循环与模式，即微交互中“不可分割的”部分，包括不同的模式和循环，以及如何使用长循环。

- 第 6 章　综合实例
  综合考虑微交互模型的各个方面，设计三个微交互示例，一个针对移动应用、一个面向在线应用，最后一个着眼于家电。这一章也将讨论如何把微交互组织起来构成好用的功能。

- 附录 测试微交互

  介绍微交互功能的测试流程。

本书宗旨

过去的十几年来，设计师一直被鼓励提出宏大的思路，解决“棘手的问题”，使用“设计思维”解决商业和政府运作中的系统性问题。没有什么问题大到设计工具不适用，从业务流程重组到城市建设规划，设计重任几乎无所不包。

设计界这种资源投放的再集中会导致什么结果，现在还不清楚。但着眼于如此宏大的规模，往往会导致人们忽略设计中的一个重要方面：令人愉快的细节！我们喜爱的产品都关注细节，比如漂亮迷人的曲线，令人自信的点击，以及与人类心智模型契合的体验。

本书要讨论另一种思维方式：撇开壮观宏大、由上而下的设计，重视自底向上、精心构思的细节。这些小地方是设计师最擅长的地方，举手之劳，就能立杆见影、活灵活现。这是改变世界的另一种方式：在看似无关紧要的时刻释放令人心动的体验！

漂亮而且起作用的细节照样带给人快乐。这种快乐既是用户的，也是设计师的，但毋庸置疑这需要技能、时间和思考来创造。这事可不简单，做好了照样能像解决大问题一样流芳百世。从用户角度想一想，谁不愿意生活里有更多的乐趣呢？

排版约定

本书使用的排版约定如下。

- *楷体*

  表示新的术语、重要的概念等。

这个图标代表小窍门、建议或说明。

这个图标代表警告信息。

使用代码

本书就是要帮读者解决实际问题的。也许你需要在自己的程序或文档中用到本书中的代码。除非大段大段地使用，否则不必与我们联系取得授权。因此，用本书中的几段代码写成一个程序不用向我们申请许可。但是销售或者分发 O’Reilly 图书随附的代码光盘则必须事先获得授权。引用书中的代码来回答问题也无需我们授权。将大段的示例代码整合到你自己的产品文档中则必须经过许可。

使用我们的代码时，希望你能标明它的出处。出处一般要包含书名、作者、出版商和书号，例如：*Microinteractions* by Dan Saffer (O’Reilly). Copyright 2013 Dan Saffer, 978-1-449-34268-5。

如果还有其他使用代码的情形需要与我们沟通，可以随时与我们联系：permissions@oreilly.com。

Safari® Books Online

Safari Books Online（www.safaribooksonline.com）是应需而变的数字图书馆。它同时以图书和视频的形式出版世界顶级技术和商务作家的专业作品。

Safari Books Online 是技术专家、软件开发人员、Web 设计师、商务人士和创意人士开展调研、解决问题、学习和认证培训的第一手资料。

对于组织团体、政府机构和个人，Safari Books Online 提供各种产品组合和灵活的定价策略。用户可通过一个功能完备的数据库检索系统访问 O’Reilly Media、Prentice Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Focal

Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redbooks、Packt、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、Jones & Bartlett、Course Technology 以及其他几十家出版社的上千种图书、培训视频和正式出版之前的书稿。要了解 Safari Books Online 的更多信息，我们网上见。

联系我们

请把对本书的评价和问题发给出版社。

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属网页，你可以在那儿找到本书的相关信息，包括勘误表、示例代码以及其他信息。本书的网站地址是：

http://oreil.ly/Microinteractions

中文版地址：

http://www.oreilly.com.cn/index.php?func=book&isbn=9787115331618

对于本书的评论和技术性问题，请发送电子邮件到：

bookquestions@oreilly.com

要了解更多 O'Reilly 图书、培训课程、会议和新闻的信息，请访问以下网站：http://www.oreilly.com

我们在 Facebook 的地址如下：http://facebook.com/oreilly

请关注我们的 Twitter 动态：http://twitter.com/oreillymedia

我们的 YouTube 视频地址如下：http://www.youtube.com/oreillymedia

致谢

非常感谢 Floris Dekker 和 Andrew McCarthy，他们是大名鼎鼎的 Little Big Details 博客的编辑和收集人，本书使用的大多数插图都选自这个博客。毫无疑问，如果不是那么方便地得到这些示例，我不可能写出这本书。除了感谢他们两位，还要感谢为他们这个博客做出贡献的人。只要我能找到名字，就会在书中提到，以示感谢。

Jack Moffett 是 Design A Day 博客的撰稿人，理所当然也要接收我的谢意。不光是因为我从他的“In the Details”主题中撷取了很多例子，更因为他对那些细节的剖析长期以来给了我很多灵感，间接导致了这本书的问世。

本书技术审稿人通过鼓励、智慧和敏锐的眼光让这本书更上一层楼，他们是 Robert Reimann、Christopher Fahey、Dani Malik、Nick Remis、Dave Hoffer、Bill Scott 和 Scott Jenson。

尽管我在 O’Reilly 出版的上一本书业绩不能算骄人，但我的编辑 Mary Tresler 和 O’Reilly 团队的其他同仁还是再次给了我机会，而且继续大力给我支持，让我完成这本关于微交互的小书。

一如往常，绝不能忽视我背后坚忍的女性（一位是人类，一位是犬科）。写本书是对我全家耐心的考验，因为我只能坐在客厅中央的高脚椅上写这本书，妨碍了大家正常的生活。我把这本书献给她们。

最后，感谢过去和当前我在学习和工作中遇到的师长和朋友。你们教给了我（有时候是强迫我掌握了）很多，让我关注细节的价值。一直以来，你们都认为细节决定体验，鼓励我把产品的细节做到极致。而这些正是本书所要揭示的。

——Dan Saffer，2013 年 2 月于旧金山

第 1 章 设计微交互

“天下没有大作品。”

——维克多·帕帕奈克（Victor Papanek）

乐队指挥中止了演奏，观众席爆发出一阵阵的怒吼。纽约爱乐乐团已经演奏到了马勒《第九交响曲》最后那舒缓、沉寂的慢板。为欣赏这场音乐会，很多观众不惜重金，几百美元买一张票。一个多小时的乐曲接近尾声，观众全神贯注地聆听着这首曲子最安静、最庄严的时刻。

就在此时，出状况了。前排某位观众那儿传来 iPhone 手机的“马林巴琴”铃声，就是那种叮叮咚咚的高调木琴的声响，而且一遍又一遍，反复不停。这是手机闹铃响了。过了一会儿，没有停。又过了一会儿，还没有停。乐队指挥 Alan Gilbert 示意停止演奏，但闹铃仍然响个不停。现场开始有观众大骂那个拿手机的人。后来乐团年长的管理者称，这个人是“赞助商 X”，而且还是一个长期赞助交响乐团的人。纽约爱乐的主场地，林肯表演艺术中心艾弗里·费雪厅，本来是要迎来一片平静和安宁的，结果却变得人声鼎沸，骂声四起。

2012 年 1 月的《纽约时报》[1] 报道，赞助商 X 一天前刚换了手机，他

注 1：2012 年 1 月 12 日《纽约时报》，Daniel J. Wakin：“Ringing Finally Ended, but There’s No Button to Stop Shame”（闹铃不响了，脸也丢尽了）。

们公司把他原来使用的黑莓换成了 iPhone。演出开始前，他把手机调成了静音，但他不知道 iPhone 的闹铃在静音模式下也会响。因此在闹铃响起时，甚至在令人难以忍受地响了很长时间后，他都没意识到是自己的 iPhone 在响。而当他意识到是自己的手机并关掉闹铃时，已经太晚了，整个演出都被搞砸了。

第二天，消息传开，引发了互联网上很多人的批评，有人也在拿这件事找乐。作曲家 Daniel Dorff 发推文说："为防万一，我把手机铃声改成了马勒《第九》。"很多博客也就此事发表意见，其中有些人赞成要静音，就禁止一切响声。比如在 2012 年 1 月份的一篇文章"Daring Fireball: On the Behavior of the iPhone Mute Switch"（关于 iPhone 静音开关的行为）中，技术专栏作家 Andy Ihnatko 写道："我认为，与其让设备错误的决定招致难堪，还不如自己干傻事儿造成难堪更好些。"

还有一些人则认为，就算静音，闹铃该响还是要响（我认为这是对的）。正如苹果专家 John Gruber 指出的（http://t.cn/zQHeaA6）："如果静音开关禁止一切响声，那么每天都将有上千人睡过头，因为他们进入梦乡的时候，可能意识不到手机处于静音模式。"

苹果的 iOS Human Interface Guidelines（http://t.cn/apfVq1）也说明了为什么静音要这样处理：

> 例如，用户在剧院中要把设备切换成静音模式，以免打扰其他观众。此时，用户仍然希望能在设备上使用应用，只是不想被意外或没有明确要求的声音（比如响铃或短信提示音）惊扰。
>
> **响铃/静音**（或**静音**）开关不会禁止用户专门或明确要求产生的声音。

换句话说，手机静音不会禁止那些用户特别要求发出的声音，而只会禁止非用户主观意愿下发出的声音（比如短信提示音、来电铃声）。这就是规则。与很多规则一样，它并不为人所知，而且除了开关上橙色的标记之外，屏幕上也没有任何指示说明铃声关闭了。假如苹果原来规定了不同的规则，即静音就禁止设备发出所有声音，那么还得为

此规定其他规则和反馈机制。闹铃不响了那手机振动吗？手机处于静音模式时，需要某种提醒装置吗，比如唤醒手机时在屏幕上显示一个图标，或者点亮一个硬件的小 LED 指示灯？设计手机静音的方式太多了。

手机静音就是微交互的一个例子。微交互，就是产品中涉及一种使用场景的交互，只体现为一种功能，只完成一件事（图 1-1 展示了一个例子）。微交互可以是启动应用，也可以启动设备，但更常见的则是伴随或内置于另一个更大的功能。微交互只需瞬间，可能没人注意，甚至没人知道，也可能妙趣横生，令人愉快。修改设置、同步数据或设备、设定闹钟、更换密码、打开家用电器、登录、设置状态消息、收藏或标注“喜欢”……都是微交互。我们随身携带的设备里，家里的各种电器内，手机或电脑的应用中，乃至我们每天工作生活的环境中，到处都可以发现微交互。

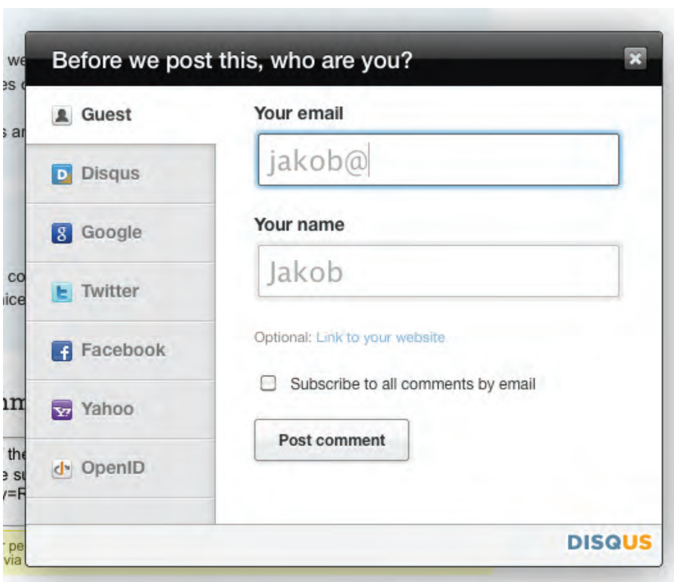


图 1-1：常见的微交互：登录。社交网站 Disqus 的登录表单能根据你的电子邮件猜测你的名字（蒙 Jakob Skjerning 和“Little Big Details”惠允使用）

微交互就是产品中这种通过交互来实现功能的细节，而正如 Charles Eames 的名言[2]所说，细节不仅仅是细节，而是设计。细节可以让用户更方便、更愉快地使用产品，尽管人们事后不会记得它们。有些微交互实际上或者事实上是看不见的，也不太可能成为你购买一款产品的理由。相反，它们通常都是一些零零碎碎的功能，或者正式一点说，属于支持性或所谓的“完整性”功能。比如说，人们不可能因为一款手机的静音功能而购买它，但这个功能却是必需的。而且，诸如此类的微交互带给用户的体验却千差万别，有的很好，有的很差。有些微交互设计得很难用，有些根本没人知道或者让人想不起来，最好的微交互都是吸引人而又聪明乖巧的。本书要讨论的就是最后一类，即怎么设计出最好的微交互。

“赞助商 X”是因微交互而声名远播的一个例子，但这样的例子并不多。正如“赞助商 X”一样，尽管我们每天都生活在各种微交互中，但往往只有在它们给我们带来麻烦之后，我们才会注意它们。然而，无论多么不起眼，多么难发现，微交互却是至关重要的。想想看，你喜欢某个产品或讨厌某个产品，往往就是因为这些产品中的某些细节。好的微交互能让我们的生活更加便捷、更有乐趣。这就是本书要讨论的主题：怎么设计好微交互？

这一章会告诉大家怎么区分什么是功能，什么是微交互，顺便简单介绍一下微交互的渊源。然后开始讨论微交互的结构，也就是本书的结构。作为一个模型，它为我们讨论和分析微交互的各个方面，进而设计好自己的微交互提供了参照。最后，我们会谈一谈微交互的设计流程。

1.1 微交互不是功能，但很重要

微观交互与宏观功能完美契合是最强大的。用户体验设计的本质就在于此：既要关注细节，又要关注大局，只有这样用户才能获得最好的体验（见图 1-2）。

注 2：见 *100 Quotes by Charles Eames*，Charles Eames（Eames Office, 2007）。

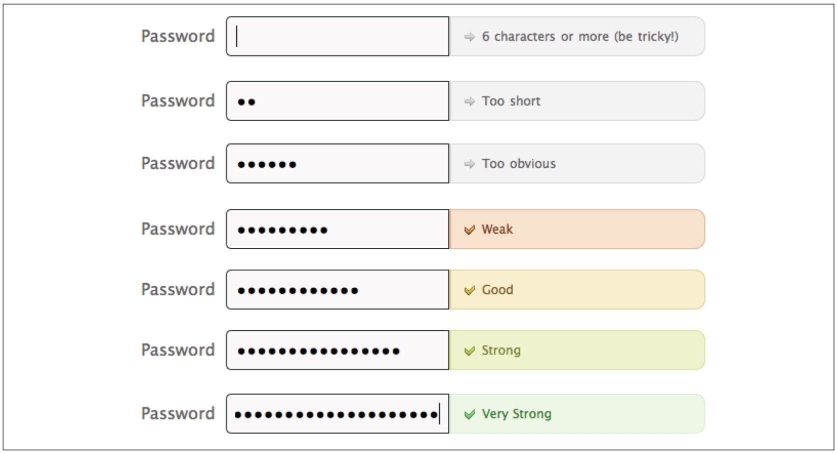


图 1-2：Twitter 的密码选择表单就是微交互的一个典型范例，反馈消息清楚明白（蒙“Little Big Details”惠允使用）

微交互与功能的区别，主要在于规模和范围。功能一般都比较复杂（多种用途），需要花较多时间熟悉，而且涉及认知的塑造。微交互一般都简单明了，认知上毫无压力（见图 1-3）。比如，音乐播放器就是功能，而调整音量则是这个功能中的一个微交互。



图 1-3：如果有人在你的 Facebook 主页上用非默认语言发表了评论，Facebook 会显示“See Translation”（查看翻译）链接（蒙 Marina Janeiko 和“Little Big Details”惠允使用）

微交互适合如下应用场景：

- 只完成一项任务；

- 连接不同的设备；
- 只影响一个数据，比如股票价格或温度；
- 控制正在进行的过程，比如切换电视频道；
- 调整某项设置；
- 查看或创建一小部分内容，比如状态消息；
- 打开或关闭某个功能。

微交互不容小视

微交互可以是产品的组成部分，也可以是整个产品。以烤面包机为例，它只做烤面包这一件事。烤面包机只有一个使用场景：把面团放进去并按启动开关。然后，烤面包机就开始烤面包。烤好之后，仓门打开。仅此而已。当然，现在有的机器有了新的烘烤目标（比如烤百吉饼）。不过从本质上说，这种设备就只是一个微交互。

类似地，小应用也可以只是一个微交互。数以千计的小应用（包括桌面和移动设备中的应用）都只专注于做一件事，有的是转换温度（比如图 1-4 所示的 Convertbot），有的是计算器，有的则显示天气预报。



图 1-4：Tapbot 的 Convertbot 是基于一个微交互构建的应用，它只专注于把一个值转换成另一个值

微交互经常是在设计和开发产品的最后阶段才被考虑的事情，因此往往不受重视。但忽视微交互是个错误。最早的 Android 手机（G1）之所以让人觉得像个半成品，原因就在于微交互做得很差劲，跟 iPhone 简直无法相提并论。比如，有很多不一致的方式可以触发删除操作，此外在某些应用中按搜索键没有任何反应。如果微交互设计得很差劲，那么无论主功能如何完美，使用该功能的过程都是让人难以忍受的。产品设计得好坏，取决于产品中最小的那一部分设计得如何。

再想一想，几乎所有操作系统，包括桌面操作系统和移动操作系统，实际上都在做同样的事情：安装和启动应用程序、管理文件、连接软件和硬件、管理打开的应用程序和窗口，等等。但操作系统之间的差别，至少从用户角度来讲，则体现在你每天甚至每小时都要与之接触的微交互上（见图 1-5 和图 1-6）。



图 1-5：本书作者 OS X 中的菜单栏，这些图标中的任何一个都对应着一个微交互

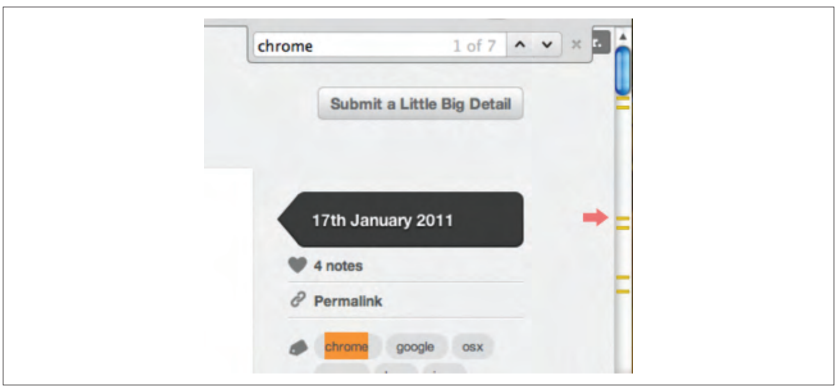


图 1-6：在网页中搜索某个词时，Chrome 会在滚动条中显示目标词所在的位置（蒙 Saul Cozens 和“Little Big Details”惠允使用）

当然，有些功能的确是特别有用或者特别强大的（或者受到知识产权法的特别保护），以至于微交互显得并不重要。很多医疗设备就是典型的例子，就像大多数技术刚刚诞生时一样，人们更多关注的是这些

技术能做什么，而不是怎么做。比如 2002 年上市的第一代“鲁姆巴”（Roomba ）家庭保洁机器人既不会计算房间大小，也不能检测障碍物和脏物，但无论如何它在当时都是新奇的技术。而后来的机型（特别是在市场上有了竞争产品后的机型）则越来越专注于人类和机器人的微交互。

在竞争激烈的市场环境下，微交互显得格外重要。如果产品的功能类似，那只能靠使用产品的体验来扩大销量并提升品牌的忠诚度。而产品的整体体验严重依赖于微交互。微交互是产品观感中的“感”。Google+ 与 Facebook 相比，之所以让人觉得单薄乏味，追根究底在于其微交互不够好。比如把用户归入圈子，一开始还让人觉得有点意思，时间一长就变得特别无聊和小儿科。

另外一个重视微交互的原因，就是在如今多平台共存的生活环境下，微交互起到了至关重要的连接作用。微交互可以把多个移动设备、电视、桌面电脑、笔记本电脑、家用电器，以及 Web 应用中的功能连接在一起。虽然这些微交互可能因平台而有所不同，但它们体积小巧，因此可以保持某种一致性，而大型的功能却做不到这一点。特别是家用电器和移动设备，它们的小屏幕（或根本没有屏幕）看起来就像是专门为微交互量身打造的一样。小巧的微交互完美契合小设备。就拿 Twitter 来说吧。Twitter 本身就构建于一个微交互：发送 140 字以内的短信。而用户几乎可以通过任何设备、在任何地方来做这件事。有些实体甚至可以独立地发推文，或者代表我们发。Twitter 可以用来传播八卦新闻，也可以成为引导革命的消息平台。设计良好的微交互能够完美地适合各种平台，适应数百万用户（见图 1-7）。

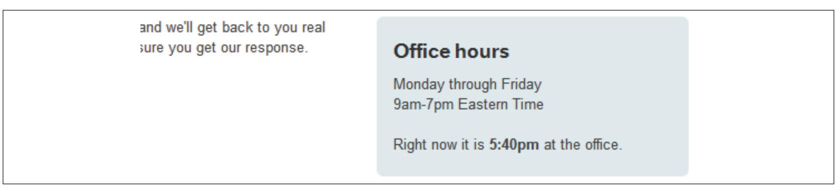


图 1-7：贴心的微消息。在 Harvest 网站上寻求帮助时，你会在他们的公布办公时间旁边看到当地目前是几点钟（蒙 Nicolas Bouliane 惠允使用）

微交互也完美契合我们拥挤、复杂、碎片化的生活。我们需要甚至乐于快速浏览数据，在餐馆迅速签到、在地铁上随意评论。（“休闲游戏”这个类别实际上就代表着一类专门针对娱乐的微交互。）

微交互强迫设计师简化工作、注重细节，要求设计师比拼谁的设计更轻巧、更简明、更流畅，而不是更笨重（见图 1-8）。

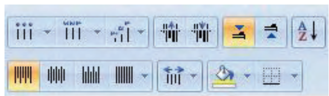


图 1-8：在 Microsoft Office 中，文本旋转之后，相关的格式按钮也会随之旋转（蒙“Little Big Details”惠允使用）

1.2 微交互秘史

1974 年，一位年轻的工程师 Larry Tesler 在为 Xerox Alto 计算机开发一个名为 Gypsy 的应用程序。Gypsy 属于第一代文字处理程序，在它之前则是具有开创意义的 Bravo。Bravo 是第一个真正“所见即所得”的文字处理程序，也是第一个具有更改字体功能的程序。虽然都是文字处理程序，但 Gypsy 却很不一样，因为它使用了鼠标和图形用户界面（GUI）。Larry 的任务是减少界面中的感觉体（modality ），以便用户不必切换到其他模式去执行操作。此后数十年，Larry 一直为这个目标而奔走呼号。你看，他的网站是 http://www.nomodes.com，Twitter 是 @nomodes，就连他的汽车牌照也是 NOMODES（http://t.cn/zQHeT28）。Larry 希望用户在按下字母键时，总能在屏幕上看到相应的字母以文本形式显示出来——这对文字处理程序而言，可不算是过分的要求。但 Bravo 就做不到这一点。在 Bravo 中，用户只能在特定的模式下输入文本，而在这个模式之外，敲击键盘会触发其他某种功能。

把文本从文档的一个部分移动到另一个部分也是文字处理程序的功能。在 Bravo 中（见图 1-9），用户必须先选择目标，然后按“I”或“R”键，进入“插入”或“替换”模式，再查找并选择要移动的文

本，最后按 Esc 键执行复制[3]。Larry 想出了一个更好的办法来执行这种操作，他的方案不仅利用了鼠标，而且大大简化了整个微交互。在 Gypsy 中，用户可以选择一段文本，然后按“复制”功能键，选择目标，最后再按“粘贴”功能键。不需要模式。剪切和粘贴就此诞生。

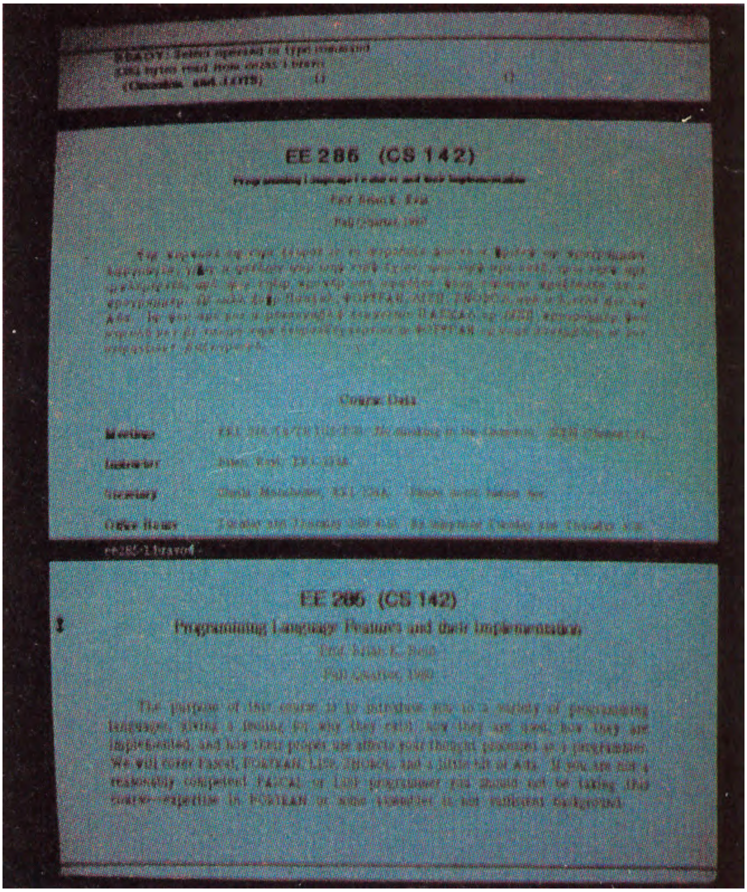


图 1-9：Bravo 的“屏幕截图”（宝丽莱照片）。通过下方的窗口在上方窗口中生成表单（蒙 DigiBarn Computer Museum 惠允使用）[4]

注 3：详见 *Bravo Course Outline*（作者 Suzan Jerome，Xerox 公司 1976 出版）。

注 4：此处翻译参考了如下链接：http://www.digibarn.com/collections/systems/xerox-alto/。

交互设计以及人机交互的历史，实际上就是微交互的历史。今天，我们在桌面和笔记本电脑中、在移动设备中习以为常的那些交互方式，都曾引发过微交互设计的革命。无论是保存文档，还是把文件拖放到文件夹，还是连接到 Wi-Fi 网络，都曾经是精心设计的微交互。就连滚动和打开多个窗口这么“基本的”交互方式，也都要经过设计。技术在快速发展，不断对微交互提出创新的要求。用户习惯了拿来就用，他们只会关注更好的交互方式，或者是那些由于技术发展催生的或强制必须使用的微交互方式。

没错，技术不断发展，支持这些技术实现的微交互同样也在不断发展。以滚动为例，Bravo 最早支持一种原始的滚动机制，但直到 Alan Kay、Adele Goldberg 和 Dan Ingalls 在 Xerox PARC 的另一个产品 SmallTalk 中引入滚动条才算基本成熟，时间大约在 1973 年到 1976 年之间。SmallTalk 支持的滚动更加平滑，可以一个像素一个像素地滚动，而不是只能一行一行地滚动。（这是 PARC 当时给史蒂夫·乔布斯以及他带来的工程师演示的一个著名的功能，后来苹果在 Lisa 以及 Mac 中都内置了这个功能，参见图 1-10。）[5]

随着文档变长，滚动条上会增加一个箭头图标，点击可以直接跳转到文档底部。此外，还会出现一个提示条，告诉你当前在文档中的什么位置。真正的变化开始于触控板和移动设备中的触摸屏技术。你现在是不是通过手指滑上滑下来滚动界面？苹果公司在发布 iPhone 之后，为了让笔记本电脑和移动设备统一成“自然滚动”，从 OS X Lion 开始改变了滚动方向（从上到下）。（可以读一读 Michael Agger 在 *Slate* 杂志上发表的文章“Apple’s Mousetrap: Why did Apple reverse the way we scroll up and down?”，链接为：http://slate.me/10nnZN8。）苹果还不惜引发众怒地隐藏了滚动条，只在滚动过程中或者在光标靠近可滚动窗口右边界时才显示它。这些都推动了微交互向前发展。

注 5：参见 Michael A. Hiltzik 所著的 *Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age*（HarperBusiness，2005）。

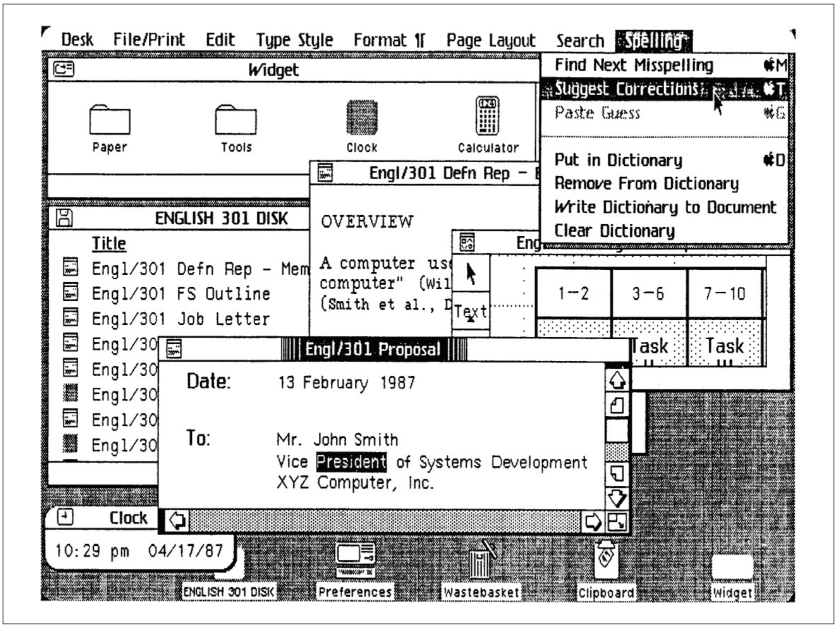


图 1-10：苹果 1982 年发布的 Lisa 个人电脑推出了一系列（从市场角度说）“新”的微交互（来源：Lisa Graphical User Interface Gallery Guidebook）

微交互并不局限于数字产品。实际上，微交互起源于第一代电子设备，比如收音机（1893 年）、手电筒（1986 年）和洗衣机（1900 年）。正像设计师 Bill DeRouchey 在他的演讲“The History of The Button”中指出的，在（电子机械出现之前的）机械时代，用户可以直接跟踪自己的操作并看到结果。比如，拉动控制杆，可以看到齿轮转动，齿轮转动又带动车轮转动。输入和输出的关联很直观。电子技术的应用改变了一切。按一下墙上的按钮，房间另一边的灯应声点亮。没错，反馈是实时的，但实现方法却不是。正如 DeRouchey 在“The History of the Button”（http://slidesha.re/1049o1K）中所说：“按钮是一个标志，标志着人类动作的结果第一次完全不同于引发这个结果的动作。”动作变得抽象起来了。

进入数字时代，特别是 GUI 出现之前，动作甚至更加抽象。插入一叠穿孔卡片，或者拨动一排开关，产生的结果几乎无法预知。终于有一

天，GUI 诞生了，微交互得以简化。但是，摩尔定律（处理器速度每 18 个月增长一倍）、库米定律（硬件的耗电量每 18 个月降低 50%）、克莱德定律（存储空间呈指数级增长），以及越来越快的网速和越来越便利的网络接入（先是局域网，后有无线网络，包括无线局域网和移动无线网），都对更多的微交互提出了需求，而这些微交互涉及的控制操作远比开灯抽象得多。比如在设备之间同步数据，这个概念本身就是抽象的，现实当中没有什么对应的类比。

输入方法同样大大改变了微交互。按键、开关、键盘、鼠标，这些传统的物理装置可以触发微交互；触摸屏、传感器、语音和手势，这些新兴的交互手段同样可以触发微交互。回首不远的过去，我们与身边环境交互的唯一方式只有通过物理装置手工调整它。从 1956 年 Robert Adler 发明 Zenith Space Commander，即第一个电视遥控器起，这种局面被彻底打破（见图 1-11），人类有史以来第一次可以不通过接触就能控制物体。



图 1-11：遥控飞机、遥控轮船早就有了（主要是军用），但让遥控器走进普通家庭的则是 Space Commander 电视（蒙 Peter Ha 惠允使用）

今天，要触发一次微交互，甚至不必在同一个房间。只要使用适当的设备，你可以在地球的另一边调整自己家里的温度（见图 1-12）。再比如，到了合适的地点，进入某个街区，你的手机就可以提醒你该做什么，或者你的 GPS 设备就可以告诉你在哪里往左拐。在公共洗手

间，只要把手伸进洗手池，水龙头就会喷出水来。你可以让自己的手机在附近找一家餐馆，可以在触摸屏上滑动手指显示出一个搜索条，可以在柜台上按一下手机买杯咖啡。这样的例子太多了。



图 1-12：Nest Learning Thermostat（学习型恒温器）通过距离传感器感知有人进入房间，然后开灯并显示温度，让人无需触碰，一目了然（蒙 Nest 惠允使用）

技术发展史的背景隐藏着微交互的发展史。微交互与技术之间的关系就像共生生物，它与技术相伴而生，如影随形，但却能影响、管理和控制技术。

1.3　微交互的结构

有效的微交互不仅表现在大小上，更表现在它们的构成上。一个设计完善的微交互必须适当地处理四个不同的部分，接下来我们会依次讨论（见图 1-13）。

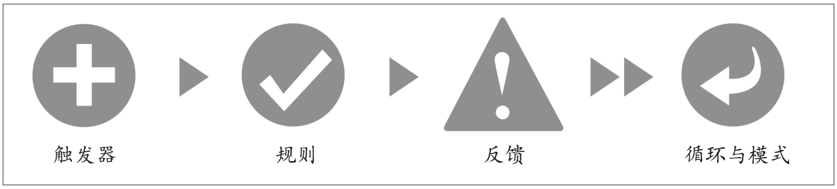


图 1-13：微交互的结构

这四个部分是设计和分解微交互的一种方式：触发器启动微交互、规则规定微交互的工作过程、反馈向用户说明规则、循环与模式是影响微交互的元规则。

任何微交互的第一部分都是*触发器*。对于关闭 iPhone 的铃声而言，触发器是由用户拨动的，也就是说用户必须要做点什么（在这里就是拨动开关）才能开始微交互。因此，很多微交互都始于对用户需求的理解：用户想要做什么，什么时候会这么想，有多频繁。用户需求决定了触发器的使用情境、操作方式和存在时间。以手机静音为例，用户任何时候都有可能想要关闭铃声，而且是快速关闭。因此，触发器（响铃 / 静音开关）必须随时待命，无论当前运行着什么应用，都需要能够立即打开或关闭声音。这个功能实在太重要了，因此 iPhone 仅有的 5 个物理按键中就有一个专门用作静音的触发器。控件，无论数字的还是物理的，都是用户启动的触发器中最重要的组成部分，它不仅能让用户发起微交互（有时候还能让用户在过程中调整微交互），而且通常还能从视觉上表明微交互还在持续（见图 1-14）。如果 iPhone 没有铃声开启 / 关闭开关，尽管你想要这个功能，也不知道去哪里找。在很多老款手机（甚至今天的很多手机）中，静音控件都被层层掩埋在了设置菜单里。就算用户知道到哪里去设置，打开或关闭铃声都需要多达 10 秒钟时间。而拨动物理开关则只需不到 1 秒钟。

当然，物理控件不一定非得是开关。尽管总能评出最佳设计方案，但再好的设计也能换种思路重新再设计一次。在 Windows Phone 中，静音触发器就是一个可以按下的摇杆按钮（也用于控制音量）。按下这个按钮，屏幕上会显示一个覆盖层，让用户选择响铃状态是“振动”还是“响铃 + 振动”。



图 1-14：触发器的例子。在 iOS（以及 Windows Mobile）中，手机锁屏时也可以使用相机。按一下相机图标，底部的滑动条就会向上抬起一段距离，表示向上滑动可以触发相机。当然，滑动解锁也可以

而且，触发器也不一定由用户启动。如果设备或应用检测到某些条件，同样可以由系统启动触发器，从而开始一次微交互。这种系统启动的触发器越来越多。触发条件多种多样，比如收到新邮件、到了预定时间，或者股票价格、用户位置变动，等等。比如手机静音，让手机与日程关联可以使它在开会的时候自动静音。或者与位置关联，让它在你进入电影院或音乐厅时自动静音。随着应用和设备的感应范围越来越广泛、环境感知能力越来越强，让它自己做决定的机会就越来越多。

第 2 章将介绍触发器。

就算不是所有用户都愿意了解能够启动触发器的各种手机元件，至少他们愿意了解这些功能的运作方式。就像“赞助商 X”一样，他可能就想知道自己的手机静音之后到底会怎么样。换句话说，用户需要了解微交互的规则。

微交互被启动之后，就会引发一系列行为。换句话说，总会发生一些事情（见图 1-15）。通常，这意味着要开始某种功能或者交互，而视觉上只会看到应用或设备的当前状态。微交互此时或许会根据数据来猜测用户想干什么。无论如何，此时都会有一条规则起作用，而这些规则通常是由设计师规定的。



图 1-15：规则的示例。使用音乐流播放服务 Spotify 并在另一个平台打开它，先前打开的 Spotify 应用会暂停。如果恢复第一个实例的播放状态，另一个平台上的实例就会暂停。这样就形成了非常一致的跨平台的体验（蒙 Sebastian Hall 惠允使用）

就拿目前最简单的微交互——开灯来说，只要一按下触发器（开关），灯

就会打开。对简单的照明系统而言，此时只有一条规则：按下开关，开灯。如果增加一个调光器或运动检测器（在检测到没有运动的物体时关闭开关），就可以修改这条规则。但“按下开关就可以开灯”这条规则实在太简单了，简单到就连一个孩子都知道怎么开灯。

如果是应用程序或者电子数字设备，那么就算是很小的微交互，其规则都要微妙得多，而且也更不容易理解。比如“赞助商 X”，他由于不了解自己手机静音的规则而导致了交响乐演出意外终止，如果他能得到反馈就好了（接下来我们会讲反馈），因为规则本身是看不见的。与 19 世纪的机械设备不同，今天的用户通常都看不到触发器启动之后的过程。（这个“特性”被黑客无数次地利用，从而让受害者在不知情的情况下把病毒安装到自己的计算机上。）

第 3 章将介绍规则。

在使用数字设备时，我们看到和听到的都很抽象。只有极少数用户知道自己在使用软件和设备时到底发生了什么。比如说，你根本就没有把“文件”放到过“文件夹”里，而你的“邮箱”里也从来没有收到过“邮件”。这些词不过是比喻，都是为了让用户理解交互过程才发明的。你所看到、听到或感受到的能够帮助你理解系统规则的东西，就是反馈，也就是微交互的第三部分。

反馈的形式可以多种多样，比如可以是视觉上的，可以是听觉上的，也可以是触觉上的（振动）。有时候，反馈会很突出也很明显，比如拨动开关时亮起的灯光；有时候，反馈会很微妙也很难发现，比如邮件应用或移动应用中的“未读”图标；有时候，在使用导航仪做好路线规划后，它会通过语音明确告诉你什么时候转弯；有时候，在某种复杂的情况下，闪烁的 LED 灯光又容易让人误解；有时候，口袋里的手机在收到短信时，会像放屁一样“噗”地一响，引来别人异样的目光；有时候，数字屏打开时的声音又像是“窃窃私语”。反馈最重要的是与动作相配，尽可能以最合适的方式传达信息。

在关闭手机铃声的例子中，让手机静音有两个反馈：一是开关在被打开和关闭时，屏幕上会短暂显示一个覆盖层；二是手机在静音状态下，开关上会显示橙色的细条。而根本看不见的，同时也让"赞助商 X"颜面扫地的，则是铃声虽然关闭，但设定的闹铃照样会响。屏幕上也没有任何提示（除了只会显示几秒钟的覆盖层之外）说明铃声已经关闭。这些都是设计权衡的结果。

触发器和反馈更重要的地方在于表达产品的个性。实际上，可以说反馈（连同整个外观一起）完全决定了产品的个性。

反馈不局限于图片、声音和振动，还可以是动画（见图 1-16）。微交互该如何出现又如何消失？物体移动时会怎么样：它跑得有多快？物体移动的方向有意义吗？



图 1-16：反馈的例子。在 Coda2 中，"Process My Order"（提交订单）按钮会在被按下时变成进度条。不过，按钮文本应该依次变成"Processing Order"（正在提交订单）和"Order Processed!"（订单提交完毕！）（蒙 Christophe Hermann 和"Little Big Details"惠允使用）

反馈也可以有自己的规则，比如什么时候出现，如何改变颜色，如何（在用户旋转手中的平板电脑时）旋转屏幕。这些规则本身也可以发展成微交互，因为用户可能希望手工设置它们。

第 4 章将介绍反馈。

微交互的最后一部分是循环与模式，它们构成了元规则。微交互随着时间推移会怎么样：交互状态要等到手工关闭才结束（就像响铃 / 静音开关那样），还是说等一小会儿就退出交互状态？有事件打断交互时或者条件变化时又会怎么样？图 1-17 展示了一个例子。

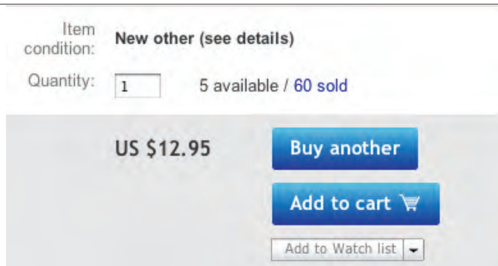


图 1-17：循环的例子。在 eBay 上，如果你曾经买过相同的商品，购买按钮就会由“Buy it now”（马上购买）变成“Buy another”（再次购买）（蒙 Jason Seney 和“Little Big Details”惠允使用）

有些微交互会包含不同的模式（尽管通常不太受欢迎）。例如一个天气应用，其主（默认）模式就是显示天气，但用户有可能必须要进入另外一个模式，才能输入地名查看某地天气。

第 5 章将讨论循环与模式。

1.4 微交互的设计理念

把微交互集成到产品中有三种方式。第一种方式是逐个设计微交互。在设计产品的过程中，或者在简化既有产品时，尽量找出所有可能的微交互。列一个微交互的单子，然后各个击破。针对每个微交互，按照本书已经介绍的结构周密考虑，争取让每个部分都有上佳表现。最终你会得到一流的微交互，可能还会创造出标志性时刻（Signature Moments）。

标志性时刻就是那些成为产品差异化标志的微交互。一个自定义的触发器控件（比如 iPod 早期版本中的滚动轮）或者一段漂亮的“加载中……”

动画或者一种好听的声音（“您有新的邮件，请注意查收！”），都可以像功能一样加以宣传和营销，而且可以跨平台使用，或者在同一品牌下不同的产品中使用。这种标志性时刻有助于提升客户忠诚度和品牌差异化。就像 Facebook 的“Like”按钮，已经成为其品牌不可分割的一部分。

各个击破方式的挑战在于如何限定每个微交互的范围。有一种将微交互转换成功能的趋势，毕竟大多数设计师对设计功能都很熟悉。我们都恨不得抓住一些大问题，把一切都解决掉。微交互就是在限定条件下做设计的实践，而且要尽可能做小。应该明确各种约束条件，专注于把一件事件做好。极简主义建筑大师密斯·凡德罗（Mies van der Rohe）的名言“少即是多”，也应该成为微交互设计师的座右铭。

第二种看待微交互的方式，是把复杂的应用程序简化为个别的产品，每个产品构建于一个微交互之上。这是一种把微交互作为产品的策略，即让产品专注于一件事，而且把这件事做好。把产品简化到极至，尽显佛性（Buddha nature）。如果你觉得需要给产品增加另一个功能，那另一个功能本身就该做成一个产品。很多家电、应用和设备，包括早期的 iPod，都遵循这个理念。这也是很多创业公司（至少是创业之初）的理念，比如 Instagram 和 Nest：他们都坚持把一件事做好。这种“最小化的可行产品”可能就是一种微交互。这种方式可以保证和驱使你从根本上简化产品，让你能够对任何增加功能的声音说不。当然，坚守这种理念不容易，特别是在那些喜欢满足用户一切需求的公司里，就更难了。想象一下：你该怎么把 Microsoft Word 简化成一个个单独的产品？实际上，很多竞争性产品正在这么做。比如，WriteApp 只负责写，剔除了文字处理程序的大多数功能，最后只专注于写，只为写作者服务。Evernote 最初也源自一个简单的微交互：跨平台记事。

还有第三种看待微交互的方式，那就是把最复杂的数字产品掰开揉碎，发现它由几十甚至几百个微交互构成。可以把产品当成所有这些微交互和谐共处的结果。这也是 Charles Eames 说细节是设计的真正含义。一切都是细节，一切都是微交互。微交互就是取悦于人的机会，微交互就是给用户惊喜的机会。正如德国著名工业设计师迪特·拉姆斯（Dieter Rams）所说：

在我内心深处，永远有一块地方为细节而保留。我认为细节比蓝图更重要。没有细节，一切皆空谈。细节是本质，是评价质量的标准。[6]

简言之，把每个功能性的细节都看成一个微交互，把整个产品看成一整套微交互的集合。这种方式下的产品设计之美，在于它反映了很多公司都大力倡导的小型化、敏捷式的工作方法（见图 1-18）。当然，缺点就是可能会迷失在微交互中，只见树木而不见森林，最终导致所有细节不能构成一个整体。这种工作方式需要更多时间和努力。

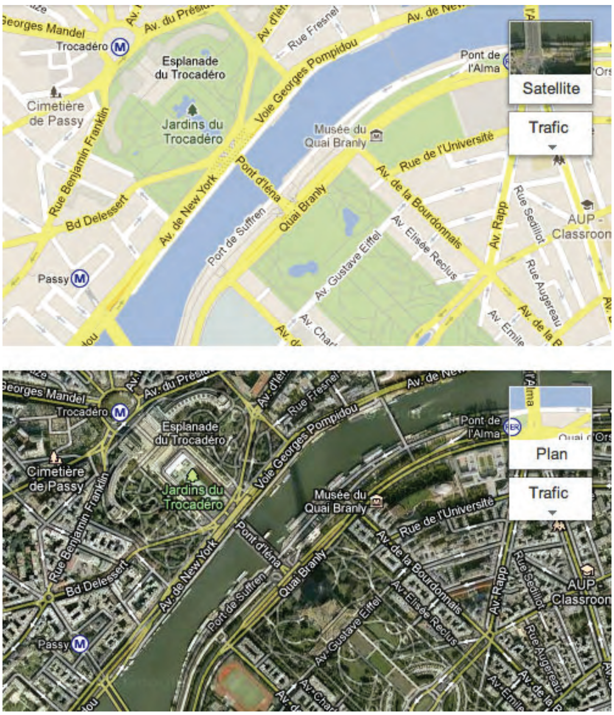


图 1-18：无论是以标准模式（“Plain”）还是卫星模式（“Satellite”）查看谷歌地图，用于切换视图的部件都会显示相应区域的地图，而且显示的是当前视图背后的另一个视图的预览（蒙 Hugo Bouquard 和 “Little Big Details” 惠允使用）

注 6：出自 Dieter Rams 与 Rido Busse 的对话（1980），*Design: Dieter Rams*（1981）中也有记载。

对于那些以快速完工而著称的公司来说，这条路并不容易走通。说实话，这种方式对任何设计师也同样都是挑战，因为客户和干系人会更关注全局，而不是那些能够增强功能或改进全局体验的小细节。没错，很难有足够的时间让你专注于微交互。要说服业务和开发团队，让他们认可在微交互设计上多花时间可不容易。对设计和开发而言，这就意味着延长工期。但这时间确实值得花。

“赞助商 X”那令人尴尬的故事提醒我们微交互很重要，而设计师的任务就是把它们设计好，否则就会给用户带来挫折和困难。Larry Tesler 在想明白可以更好地在文档内移动文本时也知道了这一点，因此剪切和粘贴才得以诞生。微交互可以让世界变得更好，每次只改变一点点。所有改变都始于触发器。

1.5 小结

微交互就是产品那些功能性的细节。专注于微交互是创建优秀用户体验的必由之路。

说起微交互的历史，可以追溯到第一台电子设备。当前数字产品很多约定俗成的交互标准，都曾是微交互领域革命的战果。

一个微交互由四个部分构成：触发器启动微交互、规则规定微交互如何起作用、反馈传达规则信息，而循环和模式构成微交互的元规则。

有三种看待微交互的方式：找到它们并各个击破、把复杂的功能简化为基本的微交互，以及把每个功能当作一组相互联系的微交互。

第 2 章

触发器

20 世纪 90 年代，纽约市公共运输局开始在公交车和地铁中推行 MetroCard（交通卡），取代自 1904 年实行的代币付费制度。当时纽约市公交地铁系统每日运送乘客 700 万人次，而 MetroCard 就是一张薄薄的像名片一样的塑料卡片。主管部门认为，分布在 5 个区的几百台自动售货机可以满足乘客购买和充值的需要。但这件事没有想象的那么顺利。纽约市常住人口超过 800 万，还有数千万人居住在三州交

界区。根据纽约市城市规划局的报告，2000 年纽约常住人口中有 36% 是国外出生的。由于不说英语的人很多，2002 年的小语种杂志和报纸就有 40 种[1]。此外，还有几万人有视力障碍、行动不便，识字很少或根本不识字，当然其中有些人兼具几种特征。据官方报告，纽约市每年有 3500 多万名游客（某些年份达到过 5000 万），其中相当一部分人会选择地铁出行，而这些人里只有少数熟悉地铁线路，或者知道怎么购买 MetroCard。事实上，纽约大都会运输署（MTA，Metropolitan Transit Authority）曾经针对 MetroCard 售卖机做过调研，用户认为其外形难看，而且操作界面也不好理解。

设计师 Masamichi Udagawa、Sigi Moeslinger，还有他们在 Antenna 设计公司的团队接受了这一挑战，担负起设计 MetroCard 售卡机的任务。

正如 Moeslinger 后来提到的[2]，她们必须排除用户有使用触摸屏自动终端的经验。那是 90 年代中期，服务业之外的人很少有机会使用触摸屏（当时的触摸屏一般在酒吧或快餐店柜台），只有一个例外：ATM 机，也就是银行的自动取款机。设计师假设用户至少都应该用过 ATM 机，但后来才知道也不是那么回事。当时，纽约公交系统的乘客中有 50% 的人根本没有银行账号，自然就没有 ATM 卡。这些人很可能怎么也不会适应 MetroCard 售卡机。“触摸屏对这些人来说是不可想象的。”Moeslinger 说。仅仅是让这几百万人了解和使用这种新机器就是一个难题。

Antenna 决定让这种机器的每个屏幕只做一件事。“它模拟对话框，每个屏幕只问一个问题。”Moeslinger 说。（换句话说，她们把每个屏幕都当成了一个微交互。）纽约大都会运输署认为这样做有问题，因为会导致每次交易时间太长。数百万人使用这种机器，每个人多耽搁几秒钟就会导致运输公司和乘客们抱怨。但不这样做恰恰更会导致这个问题。“与交易过程中显示屏幕的数量相比，能否让乘客快速地理解屏幕信息对交易速度的影响更大。”

注 1：参见 2002 年 7 月 10 日 *Editor & Publisher* 的文章“Ethnic Press Booms In New York City”。
注 2：完整的内容请见她 2008 年在 Interaction08 上的演讲“Intervention-Interaction”。

Antenna 研究了两种交互模型。一种是先放钱，再选择做什么（类似自动售货机）；另一种是先选择做什么，然后再放钱。用户更偏向第二种，而真正的问题在于怎么让用户上手使用这种新机器。

她们的方式是把整块触摸屏做成一个巨大的触发器（见图 2-1）。第 1 章提到过，触发器指的就是启动微交互的物理或数字控件。在这个例子中，整个空闲屏幕（每次交易完成后或者机器空闲时显示的屏幕）变成了一个巨大的行动召唤（call to action）：按一下开始。如图 2-1 所示，Antenna 去掉了屏幕上一切花里胡哨的东西，只专注于吸引用户看到触发器。单词“start”出现三次，而“touch”出现两次。手指动画也指向“Start”按钮。但实际上，整个屏幕都是触发器。不管你按哪里，都可以马上开始使用机器。“Start”按钮只是一个视觉提示，或者一个人造的功能点（affordance），为的只是让人知道“按”屏幕可以启动（实际上只需要轻轻碰一下）。虽然表面上看只有按钮是触发器，但实际上整个屏幕都布满了触发器。这个为应对巨大挑战而提出的解决方案非常经典，一直沿用十多年。



图 2-1：MetroCard 售卡机的空闲屏幕。Antenna 设计公司有意强调屏幕上的触发器，但跟表面上看起来不一样的是，并非只有右上角的按钮才是触发器，实际上整个屏幕都是（蒙 Antenna Design 惠允使用）

MetroCard 售卡机引入了第一条关于触发器的规则：*触发器必须让用户在使用情境下认出来其是触发器*。触发器可以是物理的控件，也可以是像 MetroCard 售卡机上的假“Start”按钮一样的按钮图像，或者是任务栏、菜单栏中的一个图标。要把它设计得让人一看就知道能通过它做什么，而且要吸引人。虽然大多数微交互都没有必要弄一个发光的手指指向开始按钮，但在其使用情境下，这种触发器是有用的，而且非常有效。

2.1 手动触发器

微交互从哪里开始呢？通常都是从用户打开设备或启动应用时最先碰到的东西开始。最早的触发器是有开和关两个状态的开关（以及功能等价的数字化界面控件）。开关和 MetroCard 售卡机的“Start”屏幕一样，都是手动触发器。（稍后我们介绍由系统发起的自动触发器。）

手动触发器通常源自用户的期待或需求：“我想打开电视。”“我想关掉手机铃声。”“我得把这一段文字从一个地方移动到另一个地方。”“我想买一张 MetroCard。”从总体来看，理解用户想要（或需要）做什么、什么时候想做，以及在什么情境下想做是非常重要的。这些问题决定了手动触发器在什么时候、什么地方出现。或许任何情况下都必须可用，比如开关按钮；或许跟使用情境紧密相关，只有各种条件具备的时候，比如进入特定的模式或者进入应用的特殊功能区域时才出现。Microsoft Office 的“迷你格式化工具条”只在你选中文本时才会出现。理解用户需求的方式有很多，借助设计研究（观察、访问、演习）或凭借对相关领域的直觉和理解去体察是常见的方式，而等到产品测试、产品发布或上市时再认识到用户需求就为时已晚。所以，让触发器的位置跟用户需求（何时、何地）相吻合是非常关键的。（参见 2.1.2 节的“手动触发器要引人注目”。）

触发器的第二条规则：*保证触发器每次都触发相同的动作*。这还用得着说吗？有读者可能会感到不可思议。事实上，唯有如此才能保证用户对微交互形成准确的心智模型。而且，现实当中违反这一条的现象不胜枚举。技术评论家 David Pogue 曾这样评论过 Samsung 的笔记应用 S Note：

S Note中的某些图标不同的时候按会显示不同的菜单。我可不是瞎说的[3]。

还有一个例子就是 iPhone 和 iPad 的主屏幕按钮，按这个按钮会返回主屏幕；如果本来就在主屏幕上，那么就会转到搜索屏幕。（暂且不说按两次或按住不放又会执行什么动作。参见 5.1 节中“弹簧模式与一次性模式”。）虽然在一个按钮上绑定多个功能是为了重用有限的硬件，但在主屏幕状态下按一次就跑到搜索屏幕恐怕也太离谱了，这时候应该什么也不做才对（或者给出“嘿，您已经在主屏幕了！”之类的提示）。

说起效果最差的可见触发器，恐怕要数下拉菜单中的菜单项了。菜单项因为在菜单里，所以实际上是不可见的。它们对应的微交互一般都不常用，而把它们隐藏在菜单里会给用户查找带来一些不便。当然，把一个不常用的微交互的触发器直接放到屏幕上似乎也不太好。“设置”就是一个典型的例子，用户不会每天都用到它，但它对某些应用又非常重要（比如图 2-2 中的桌面就是通过“设置”决定的）。因此，设计这样一个触发器，特别是权衡其可见性从来都是个设计难题。

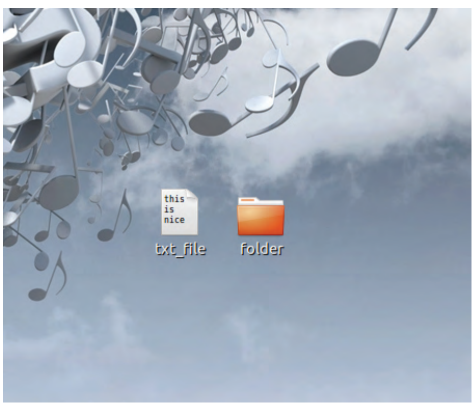


图 2-2：在 Gnome 桌面上，文件图标不是静态的，而是会显示内容的前三行（蒙 Drazen Peric 和“Little Big Details”惠允使用）

注 3：参见 2012 年 8 月 15 日《纽约时报》的报道“A Tablet Straining to Do It All”（http://nyti.ms/17Szx8y）。

2.1.1　提前展示数据

手动触发器的第三个规则是*提前展示数据*。触发器本身可以反映微交互中包含的数据。问问自己，在微交互开始之前或者启动过程中，有什么内部状态可以展示？什么信息最有价值，可以在这时候展示出来？要回答这个问题，必须知道大多数用户使用微交互干什么，而且必须在微交互开始之前就要掌握那个关键的信息。比如股市应用，它可以通过颜色或箭头表示当前市场或股票的状态，提醒用户是否有必要启动微交互。此时的触发器本身就是一个背景信息，用户扫视时注意到它就可能启动微交互。

触发器也可以表示产品中任务的进度（见图 2-3）。比如启动烤面包机的按钮时，它应该告诉用户烤好面包需要多长时间。



图 2-3：谷歌的 Chrome 浏览器图标（启动浏览器的触发器）也会显示当前的下载任务和下载进度

2.1.2　触发器的构成

手动触发器包含三个部分：控件、控件状态、文本或图示标签。

1. 控件

手动触发器至少要有一个控件（见图 2-4）。至于选择什么样的控件，取决于你想给出多少个控件。

- 对于一个动作（例如“快进”），一个按钮或简单手势足矣。这里的“按钮”可以是图标或菜单项，而手势可以是轻击、滑动或摇动。当然，按钮同样也可以是（或者搭配）一个键盘命令或一个手势。

- 对于有两个状态（例如“开”或“关”）的动作，一个拨动开关足矣。拨动按钮当然也可以，但问题是按钮很难表现状态，因此不能让人一目了然，甚至用户不会想到按钮还有另一种状态。第三种（或许也是最差的）可能是使用常规按钮，按一下改变状态。如果你想使用这种方法，那应该保证按钮状态绝对清楚。灯本身就有清楚的开关状态，因此常规（非拨动）按钮可以用于控制灯的打开和关闭。
- 对于有多个状态的动作，可以使用拨号盘。除了有一个止动装置，拨号盘还有一个推 / 拉状态。另外，也可以使用一组按钮，每个按钮代表一个选项。
- 对于在一定范围内连续进行的动作（例如调整音量），滑动条或拨号盘（特别是能够快速旋转的 Jog Dial 转点通滚轮）是最佳选择。另外，特别是在没有一定范围的情况下，可以使用两个按钮来改变值的大小或高低。
- 有些手动触发器由多个控件或表单字段（单选按钮、复选框、文本输入字段）之类的元素组织。比如，登录这种微交互就需要文本输入字段来填写用户名和密码。多个控件组成的触发器要尽量少用，如果要用，最好也要能自动填充之前输入过的值或者智能的默认值。

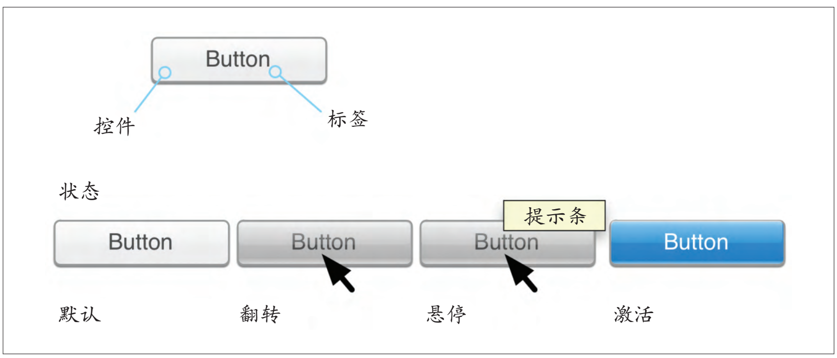


图 2-4：控件的组成部分

除了按钮、开关、拨号盘等传统控件之外，还可以有自定义的控件，比如最早（非触摸）iPod 上的滚轮。自定义控件可以让微交互显得别具一格，甚至可以使微交互成为标志性时刻。自定义控件也可以是手

势或触摸方式（参见本节后面“不可见的触发器”）。

微交互的目标是使选项最少，而不是提供智能默认值和非常有限的选项。选作触发器的控件应该体现这一理念。

控件与视觉使用情境（visual affordance），即用户期待能看到什么，紧密相关。触发器的第四条规则就是*不要破坏视觉使用情境*。换句话说，如果触发器看起来像按钮，那它就应该像按钮一样能被按下去。

手动触发器要引人注目。关于触发器的最重要问题是：触发器要有多引人注目？触发器的第五条规则是*用得越多的触发器越要引人注目*。畅销书作者 Scott Berkun 对引人注目这一点有过精彩的评述，以下是我移植到微交互的版本：

> *多数人经常要用的微交互，应该最引人注目。*
> *少数人有时会用的微交互，应该容易注意到。*
> *极少数人不常用的微交互，应该通过搜索找。*[4]

这是判断触发器需要多引人注目的黄金规则。

可是，我们到底怎么发现事物呢？

人类注意周围环境中事物的情况分两种。第一种，周围的事物如果正在移动或发出了声响，会让人不自觉地注意它们。这种对于刺激的反应是我们祖先生存的本能，正因为有这个本能，他们才会注意到周围的犀牛和其他危险动物。设计师可以利用这一点，通过移动和声响来吸引人注意触发器。但这样做有时会令人讨厌，特别是在桌面或网页中。由于人类对移动和声响会不自觉地关注，因此较高优先级的微交互的触发器才可以使用这种方式。而能够不断这样做的，只有那些最高优先级的微交互，比如出错和警告。

人类关注事物的第二种情况是主动搜寻，即有目的地寻找。我们会盯住某些东西或区域，希望从中找到满足自己需要的内容。只要视力没

注 4：改编自 Scott Berkun 的文章“The Myth of Discoverability”（http://t.cn/zHrpa0k）。

有问题，这种发现事物的方式是最直观的。我们会运用身体、大脑，或者只用眼睛去寻找自己想要的东西。

> 要注意的是，人类对听觉的反应要快过对视觉的反应。听觉刺激只需 8~10 毫秒即可传达到大脑，而视觉刺激则需要 20~40 毫秒。[5] 人类对听觉的反应需要 140~160 毫秒，而对视觉的反应需要 180~200 毫秒，同样是前者更快。[6] 人眼的视角是水平 180 度，垂直 100 度，而听觉则是 360 度的。我们的祖先看不到来自身后的捕食者，但却可以听到。（某些爬行动物和鸟类的视角则也是 360 度。）虽然理论上可以使用某种声音来帮助用户定位触发器，但在实践中则很难实行。

我们在寻找东西时，视野会变窄，甚至达到 1 度[7] 或者只有平常视野的 1%。这么窄的视野堪比聚光灯[8] 和放大镜[9]。此时，我们专注于辨别物体，即对环境中的事物进行识别和归类。

我们在辨别物体时，眼睛会寻找熟悉的小几何体（geon）形状。所谓小几何体，就是正方形、三角形、立方体、圆柱体等简单的几何形状，我们的大脑把它们组合起来就能辨别出物体。[10]

注 5：Marshall,W. H.、S. A. Talbot 和 H. W. Ades.“Cortical response of the anaesthesized cat to gross photic and electrical afferent stimulation.” *Journal of Neurophysiology* 6: 1–15. (1943)。

注 6：Welford, A. T.“Choice reaction time: Basic concepts.”（选择反应时间：基本概念）A. T. Welford (Ed.), *Reaction Times*. Academic Press, New York, pp. 73–128. (1980)。

注 7：Eriksen, C; Hoffman, J.“Temporal and spatial characteristics of selective encoding from visual displays”（视觉显示选择性编码的时空特征）*Perception & Psychophysics* 12 (2B): 201–204. (1972)。

注 8：同上。

注 9：Eriksen, C; St James, J.“Visual attention within and around the field of focal attention: A zoom lens model”（注意领域之内及周围的视觉集中：变焦镜头模型）*Perception & Psychophysics* 40 (4): 225–240. (1986)。

注 10：小几何体最早见于“Recognition-by-components: A theory of human image understanding”，作者 Irving Biederman，刊载于 *Psychological Review* 94 (2): 115–47. (1987)。

既然人眼以小几何体为基本识别单位，那么一些图标形触发器就特别适合设计成某种几何形状。一般来说，只具备一种特征的目标比具备多种特征的目标更容易被找到。[11] 因此，要尽量保证触发器看起来简单，尤其是在复杂拥挤的环境下（比如混在一大堆图标中间）。

辨别出某个物体之后（“那是个按钮”），我们一般会将其与对应的使用情境关联（“按钮可以按”），除非还有其他视觉提示表明使用情境不适用，比如变灰或上面放个大红叉（X）。手动触发器的第六条规则就是*不要引起用户对使用情境的错觉*。如果有一个图标看起来像按钮，那么它就应该表现得像按钮一样。微交互的目标就是把识别成本降到最低，千万别让用户猜你的触发器怎么工作。因此要尽量使用标准的控件。正如 Charles Eames 所说：“创新是没有办法的办法。”

最引人注目的触发器依次是：

- 移动的物体，比如脉动图标；
- 带使用情境和标签的物体，比如带标签的按钮；
- 带标签的物体，比如带标签的图标；
- 只有一个物体，比如图标；
- 只有一个标签，比如菜单项；
- 什么也没有——不可见的触发器。

不可见的触发器。手动触发器也可以是不可见的。换句话说，就是没有标签，也没有使用情境让用户知道怎么去触发相应的微交互。不可见的触发器通常都离不开传感器，比如触摸屏、摄像头、麦克风、加速计等（见图 2-5）。此外，不可见的触发器也可以只是一个命令键（见图 2-6）或鼠标移动（比如，移动到屏幕一角）。

注 11：Treisman, A.“Features and objects in visual processing”（视觉处理中的特性与对象）《科学美国人》, 255, 114B–125. (1986)。

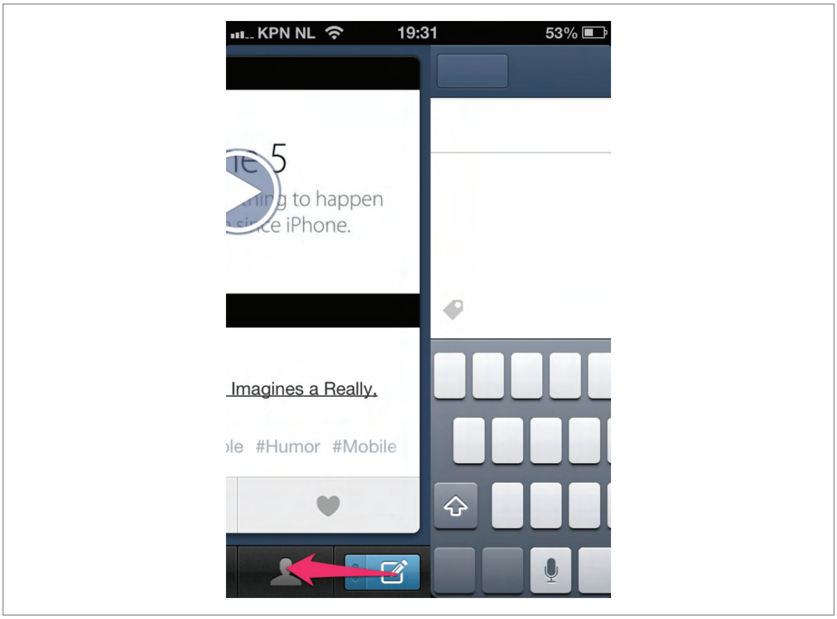


图 2-5：在 iPhone 手机的 Tumblr 应用中，手指向左扫过（而不是按下）按钮，可以创建一篇新博客；向上扫过按钮则可以创建一篇新图片博客（蒙 Robin van't Slot 和“Little Big Details”惠允使用）



图 2-6：在 Alfred 的设置中，如果禁用可见的触发器，则不可见的触发器也会被选中（蒙 Hans Petter Eikemo 和“Little Big Details”惠允使用）

当前的触摸屏用户界面隐藏着最多的不可见控件。没有可见的使用情境表明各种多点触摸手势的存在，而轻击和滑动之外的自定义手势则往往要通过试错来发现（见图 2-7）。

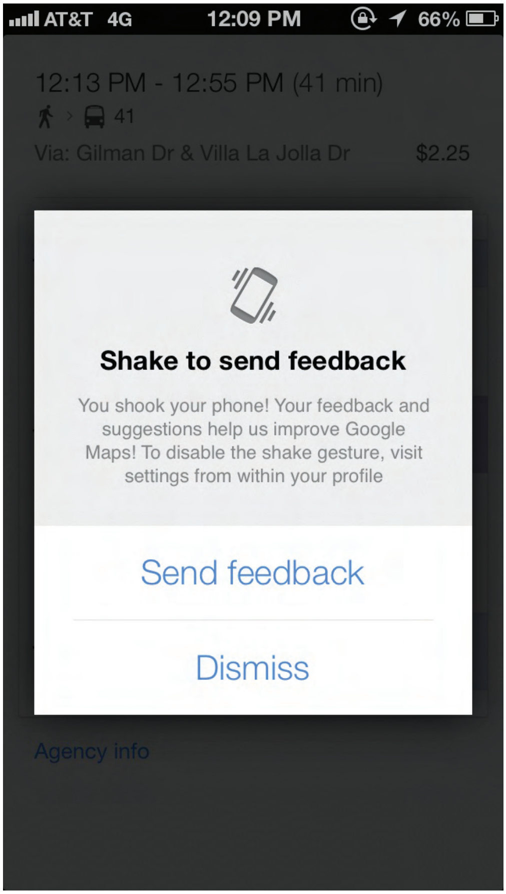


图 2-7：在 iOS 系统的 Google Maps 中，摇晃就是一个发表反馈的不可见触发器（蒙“Little Big Details”惠允使用）

语音输入也是不可见控件的例子，有三种语音输入。

- 始终监听

  产品的麦克风始终开启，用户只要对着它说出指令即可（通常要提到产品的名字）。微软 Kinect for Xbox 就属于这种情况，其“Xbox, play!”就是始终监听控件的典型例子。

- 对话

  产品的麦克风只在特定的时候开启，以监听对某个提示的响应。（“要继续使用英语，请说‘yes’。”）自动化程度最高的客服电话就属于这种情况。

- 与控件配合

  为发出语音指令，必须有某个物理控件配合。苹果的 Siri 大致就是这样：用户必须按住主屏幕按钮才能发出语音指令。

另外，摆手或者摇晃等触发器通常也是不可见的。比如语音控件，有时要使用手势必须先有一个初始动作（比如摆手）或通过某个物理控件让设备就绪。配戴谷歌眼镜时，向上昂头或触摸一下镜架即可开启屏幕。触摸或接近设备也可以作为不可见触发器，比如把手伸到水龙头下就会自动出水。类似地，远离物体也可以是触发器，比如马桶在人走之后会自动冲水。

为什么还会有不可见触发器呢？事实上，无论什么交互界面都不可能保证所有东西能够被立即发现。让一切可见经常会导致屏幕上杂乱无章，没有头绪。隐藏一些东西可以让屏幕或物体看起来更简单，同时还不会舍弃功能（见图 2-8）。不可见控件能够起到强调可见控件的作用，让重要和不重要控件得以区别。不过要知道，不可见并非微交互（以及任何交互）设计过程中直接追求的目标，而是上下文和技术的副产品：在这种情境下，最好把什么隐藏起来？或者基于技术条件限制，如果不把什么隐藏起来，可见控件就没有地方显示？最好的微交互应该具有恰到好处的界面，而不是控件越多越好。

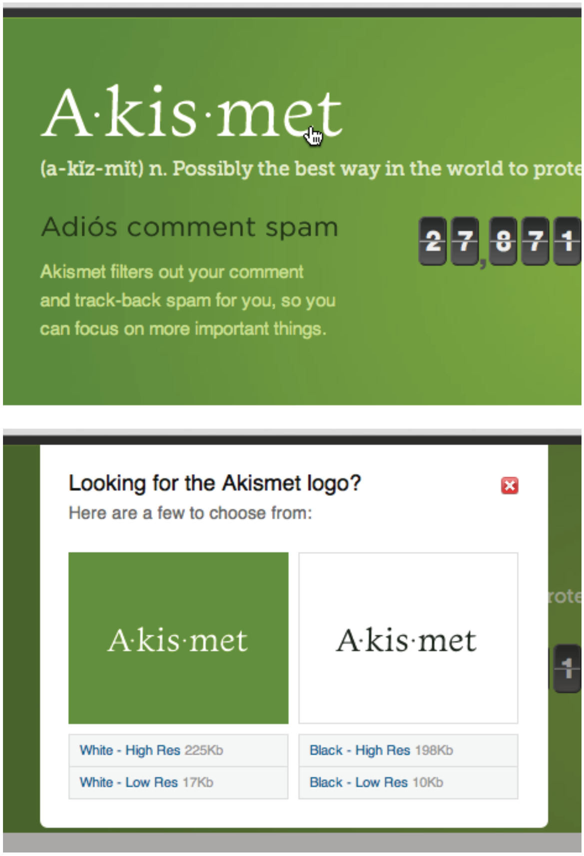


图 2-8：Akismet 有一个合适的不可见触发器。在用户右击其标志图片时（有可能是想要保存这张图片），Akismet 就会显示一个窗口，提供几种不同分辨率的图片供下载（蒙 Fabian Beiner 惠允使用）

不可见触发器应该能够习得。发现微交互之后（不管是意外发现、听别人说，还是查帮助学会的），用户经常只能依赖于自己（有问题）的记忆再次将其启动。能够习得意味着不可见触发器应该随处可用，或者退而求其次，必须在特定条件下是可用的（见图 2-9）。不可见触发器应该容易猜到，或者说，在理想情况下可以让用户在执行其他动作时偶然发现。比如，向上滚动到列表顶部，显示“重新加载”微交互。



图 2-9：KanaSwirl 的设置中支持禁用不可见的触发器（Shake to Pause，摇晃暂停）（蒙 Shawn M. Moore 和“Little Big Details”惠允使用）

除非真没有地方或屏幕空间放置可见控件（比如谷歌眼镜），否则绝不要把不可见触发器用于最高优先级的微交互。无论如何，至少都要为微交互创建一个可见的触发器，比如命令键或菜单项。

2. 控件状态

有些手动触发器有多个状态。作为用户，不一定能全都碰上这些状态，但作为设计师，则应该全部考虑到。

- 默认状态
  不活动的空闲状态。

- 活动状态
  如果后台有活动（比如下载、更新或同步），可以通过触发器来表示。

- 悬停状态
  可以给出一个提示条式的说明、扩展触发器的大小以展示更多控件或表单字段，或者就简单地显示为当前控件是可以点击的。更进一步，还可以通过悬停展示微交互中的数据（见图 2-10）。比如，悬停在天气预报应用的图标上时，不用启动应用就可以提前展示数据，显示当前的天气。



图 2-10：在 Rdio 播放器上，悬停在快进和快退按钮上会显示下一首和上一首歌曲（蒙 Nicholas Kreidberg 和“Little Big Details”惠允使用）

- 翻转状态
  经常用于表示存在或活动，或者只添加一个指示器，表明光标位置正确，可以继续操作（见图 2-11）。

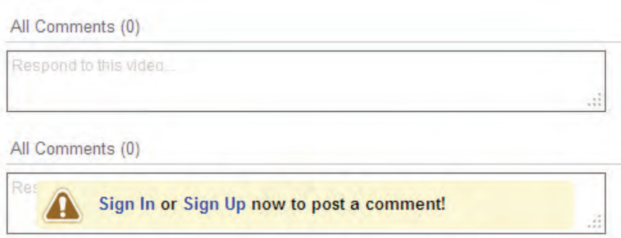


图 2-11：未登录时点击评论输入框，YouTube 会提示登录或注册（蒙 Marian Buhnici 和“Little Big Details”惠允使用）

- 鼠标单击/手指轻击/进行中

  鼠标单击、手指轻击触发器，或者触发器正在启动微交互。此时，可以让触发器消失、打开、改变颜色，或者变成表示微交互加载过程的进度条（见图 2-12 到图 2-14）。另一种可能是触发器不立即启动微交互，而是扩展显示更多控件。比如，点击“保存”按钮可以打开一个窗体询问是“覆盖”还是“另存为”。



图 2-12：单击 Path 的 Sign Up 按钮会出现笑脸图标（蒙“Little Big Details”惠允使用）

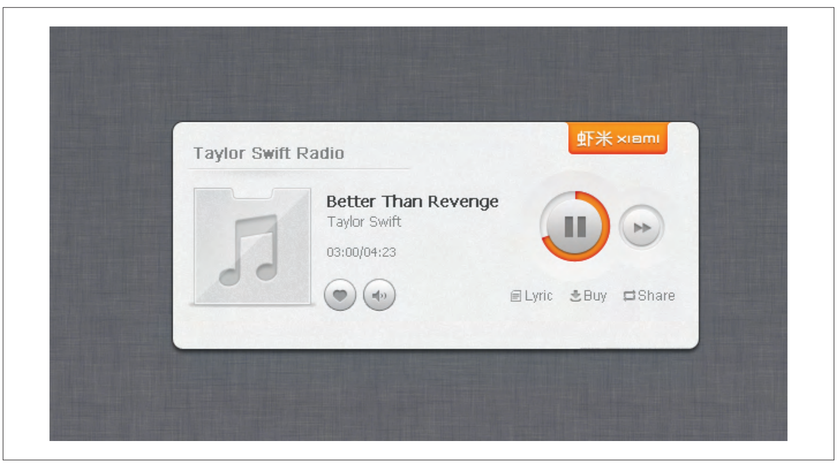


图 2-13：虾米的播放 / 暂停控件表示一首歌的播放时间（蒙“Little Big Details”惠允使用）

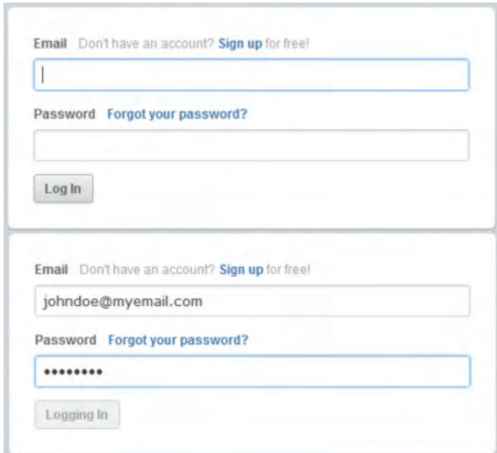


图 2-14：在 CloudApp 中，登录按钮在被单击后会改变状态，让用户知道后台已经开始处理了（蒙“Little Big Details”惠允使用）

- 切换状态

  开关和按钮可以表明自己当前的状态（左 / 右、上 / 下或按过 / 未按过）。实体设备上的开关可以让人一目了然，按钮则需要有其他搭配的指示装置，比如用发光 LED 表示按钮被按下。

- 设置状态

  拨号盘、开关和滑动条可以显示微交互当前的设置和阶段（见图 2-13）。

以上这些状态通常由触发器改变外观或展示动画来表示，但有时候也可以通过指示灯（如触发器旁边的 LED 灯）来表示。比如，开关旁边的红光 LED 可以用来表示设置关闭。切记，跟触发器不是一体的状态指示器一定要放在触发器旁边。扩展触发器时也一样：不要随处打开窗口，要把焦点放在触发器上。

3. 标签

对某些触发器而言，标签也是非常重要的部分。标签可以是微交互的名称（例如菜单项或微软功能区的名称），也可以作为状态指示器，

比如拨号盘每个止动装置的名称。标签就是界面。

标签的作用显而易见：我是不是在做自己想做的事？标签说明动作，帮助消除歧义。不过，鉴于标签会增添触发器的复杂性，需要额外的关注和思考，因此若非担心引发歧义，不要使用标签。最好的结果当然是把控件设计得毫无歧义（见图 2-15）。

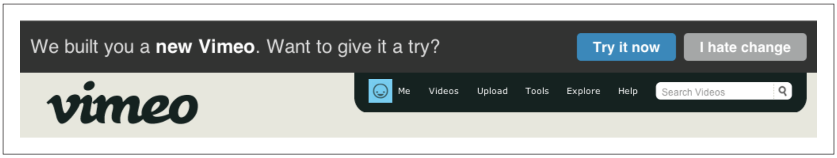


图 2-15：Vimeo 的表示“取消 / 待会儿 / 现在不行”的按钮幽默地写着“I hate change”（我讨厌改变）（蒙 Joe Ortenzi 和“Little Big Details”惠允使用）

手动触发器的第七条规则是：*仅在触发器本身无法提供相应信息的情况下才使用标签*。想一想，如果不添加，那在视觉上如何表现标签？比如，要设计一个 1~5 星的评分系统，可以采用带有 1~5 标签的滑动条，也可以让触发器带 5 颗星星，然后随着鼠标悬停，一个一个地点亮。

不使用标签在某些情况下是不可能或不合适的。一个没有标签的按钮很难与周围的其他按钮区别开来，因此很难获得点击。

与其他产品相关的（培训性或营销性的）方案不同，微交互不是品牌创意的用武之地；相反，微交互非常注重实用性，目的就是清楚明确地传达功能（见图 2-16 和图 2-17）。这并不是说微交互不重视奇思妙想或个性设计，而是说前提必须确保标签清楚明确。谷歌“手气不错”按钮的标签很有意思，但它却没有告诉你点击按钮之后会发生什么。因为没有事前反馈，不能在事件发生前就让人知道会出现什么结果。[12]

注 12：关于事前反馈的更多信息，可参考“But how, Donald, tell us how?: On the creation of meaning in interaction design through feedforward and inherent feedback”，作者 Tom Djajadiningrat、Kees Overbeeke 和 Stephan Wensveen, Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques, ACM, New York, NY, USA (2002)。

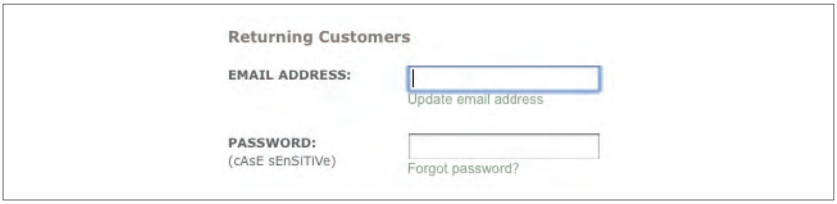


图 2-16：巴诺书店（Barnes & Noble）的网站在密码字段上使用标签表明区分大小写（蒙 Paul Clip 和“Little Big Details”惠允使用）

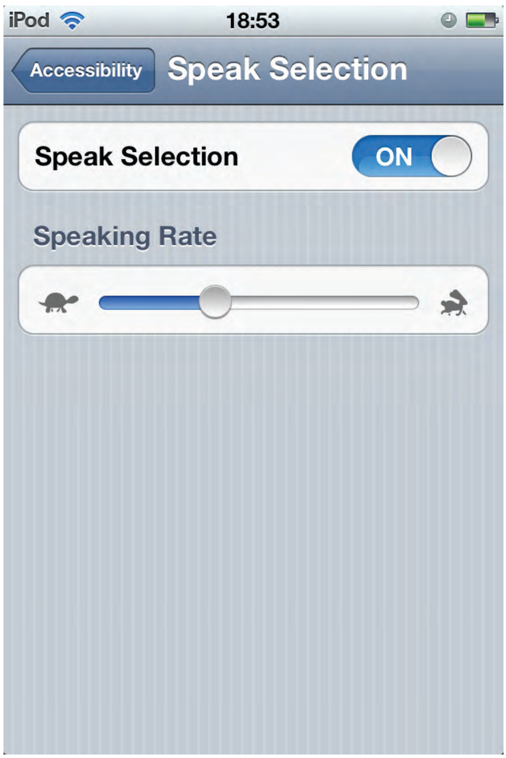


图 2-17：苹果 iOS Speak Selection 设置界面上有一个新奇但明确的图示标签，它借用了“龟兔赛跑”的寓言。然而，在文化背景中没有这个成语的地方，这两个图示可能会让人感到迷惑（蒙 Victor Boaretto 和“Little Big Details”惠允使用）

一般来说，标签应该简洁明了（见图 2-18）。“提交”作为按钮标签可能够简洁，但在非技术情境中恐怕意思并不明确。在微交互中，特指度（specificity，或具体程度）非常重要。含糊其辞是标签最大的忌讳，标签必须明确。（关于这一点，请参见第 3 章 3.3 节“微文案”。）



图 2-18：iPhone 的“滑动解锁”触发器会随着滑动而逐渐消失（蒙“Little Big Details”惠允使用）

一致性同样重要。由于标签可能是名称，所以一定要保证所有地方（微交互本身、状态、设置、数据）的标签名称统一。不要这里叫“警告”，那里又变成了“警示”。

把标签用好的根本就是要以使用者的语言来撰写它。如果标签中包含

技术术语，那用户也应该是技术人士；否则，不如用随意的大白话。另外，还要让目标用户测试标签（参见附录）。毫不夸张地说，大多数可用性问题都是由于滥用标签（或根本没有标签）导致的。

2.2 系统触发器

并非所有触发器都是手动的。事实上，当前我们身边的触发器大都不是用户而是系统发起的。系统触发器无需用户介入，只要满足条件（一个或多个）就会自动触发，如图 2-19 和图 2-20 所示。



图 2-19：发货应用 Deliveries 会在启动时检测剪贴板中是否有快递单号，如果有则通过系统触发器启动微交互。而且，它还能够智能地指出单号来自哪个快递员（蒙 Patrick Patience 和“Little Big Details”惠允使用）

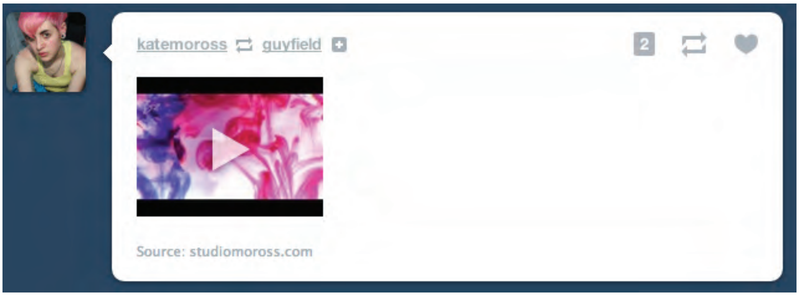


图 2-20：由他人引发的触发器。在 Tumblr 上，当你关注的某个人提到另一个你没有关注的人时，会出现一个关注按钮（蒙 Brian Jacobs 和“Little Big Details”惠允使用）

系统触发器启动的常见条件如下。

- 错误

  系统出错时，经常会通过微交互来解决问题，比如询问用户想怎么办，或者只是显示一条消息说明出了什么毛病（见图 2-21）。

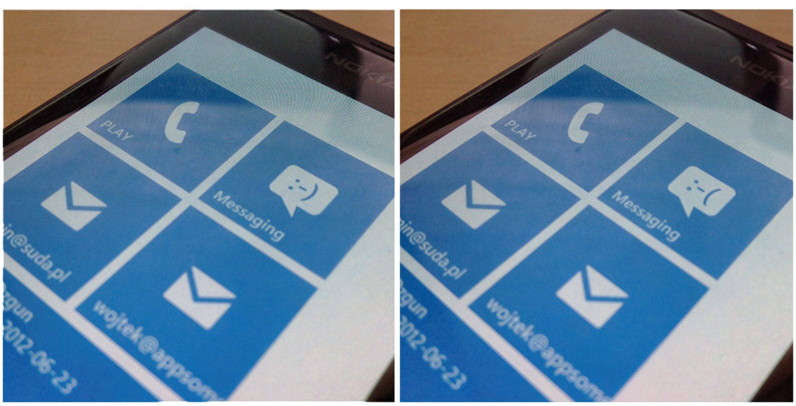


图 2-21：在 Windows Phone 中，如果发送短信失败，则短信图标（触发器）会由笑脸变成哭脸（蒙 Wojtek Siudzinski 和“Little Big Details”惠允使用）

- 位置

  位置可能是各种尺度：国家、城市、街道，甚至特定的房间。满足其中任何尺度的用户都可能触发微交互。

- 收到数据

  电子邮件、状态消息、软件更新、天气预报、亮度变化等进入联网设备和应用的数据，都可能触发“您有一封新邮件！”之类的微交互。

- 内部数据

  时间或系统资源等内部数据也可能是触发器启动的条件（见图 2-22）。比如，过一定时间屏幕自动降低亮度。

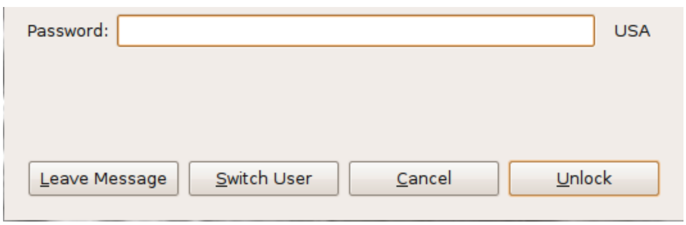


图 2-22：在 Ubuntu 中，如果屏幕因设定时间到而锁定，会出现一个让访问者给所有者留言的触发器（蒙 Herman Koos Scheele 和“Little Big Details”惠允使用）

- 其他微交互

  这是一种特殊的触发器，由其他微交互触发。最简单的例子就是向导式界面：第一步（微交互）结束触发第二步（另一个微交互），依此类推。（参见第 6 章 6.5 节“编排微交互”。）

- 其他人

  在诸多社交关系中，其他人的行为（例如，回应聊天、发表照片或消息、申请成为朋友），也可以成为触发器。

虽然用户不必手动启动这些触发器，但最好是（比如在“设置”里）提供选项让用户能够调整它们。系统启动的每个触发器都应该有手动

管理及禁用它们的选项。理想情况下，应该在微交互已经触发的时候给出这些选项（“不再显示这些提示”），至少有相应的设置。

此外，即使存在系统触发器，用户也可能需要一个手动控件（见图 2-23）。比如，用户可能需要手动同步文本，而不是等待系统自动同步。手动控件可以提供一个保障，同时也能在系统出错（网络连接断开或传感器未指示）时保证能够触发微交互。

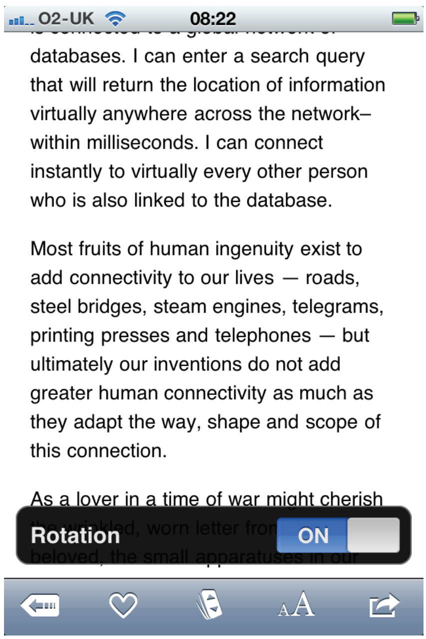


图 2-23：在 iPhone 的 Instapaper 应用中，如果你不小心把竖屏变成横屏，然后又很快转回去，就会出现一个 Rotation（旋转）锁定控件（蒙 Richard Harrison 和“Little Big Details”惠允使用）

系统触发器的规则

某些系统触发器本身需要遵循自己的规则，其中最常见的就是启动时间和启动频率（见图 2-24）。太过频繁地读取传感器数据或持续连接

远程服务器，可能会消耗过多的系统资源，比如耗费电量、占用带宽或处理器时间。



图 2-24：导航应用 Waze 知道我在傍晚打开它时可能正在开车回家，于是就给出了这个选项

系统触发器的规则需要回答以下问题。

- 这个触发器多长时间启动一次？
- 已经知道哪些与用户相关的数据？怎么利用这些数据才能让触发器效率更高、更令人愉快或更能体现用户个性？比如，知道用户当地时间是午夜时分，就可以减少系统触发器启动的次数。（参见第 3 章 3.1.5 节“不要从零开始”。）
- 有没有触发器已经启动的指示器？当时是否给出了可见的状态更新？启动指示器前后有何变化？

- 系统出错时（例如网络连接断开、数据无效时）会发生什么？不再尝试启动，还是再次尝试？再次尝试的间隔时间多长？（第 5 章还会更全面地介绍循环。）

系统触发器的规则与整体的规则紧密相关，具体将在第 3 章讨论。

与 MetroCard 售卡机的开始屏幕一样，最好的触发器应该符合情境、体贴用户。触发器的控件应该与其要传达的状态匹配，且引人注目的程度也要与其使用频率匹配。标签必须简洁明了，通俗易懂。最重要的是，触发器要能把用户带到实际的交互中，这是一切规则的前提。

2.3 小结

触发器就是任何能够启动微交互的东西。手动触发器是由用户发起的，可以是控件、图标、表单、声音、触摸或手势。系统触发器会在某个或某些条件具备时触发。

触发器必须能在特定上下文中让用户联想到它是触发器。同样的触发器每次都应该执行同样的操作。

别忘了提前展示数据。只要有可能，就要通过触发器来展示微交互内部的重要信息，比如未读短信或操作进度。

如果触发器看起来像按钮，那么它就应该像按钮一样可以被点击。不要破坏视觉使用情境。

使用频率越高的微交互，其触发器应该越引人注目。最差也要作为菜单项摆放触发器。

在触发器无法传达所有必要信息或者有必要消除歧义时，可以使用标签。标签应该使用通俗直白的语言，而且要简洁明了。

系统触发器何时触发，以及多长时间触发一次，都要遵循规则。

第 3 章

规则



2010 年 10 月，苹果在“Back to the Mac”大会上宣布了当时最新版的桌面操作系统 Mac OS X Lion（10.7 版），但直到 9 个月后的 2011 年 7 月才真正发布。系统发布当天就销售 100 万套，随后又卖出 600 多万套。在这个操作系统中，苹果内置了最新版的 Calendar、Mail 和 Address Book 应用。与此同时，也有一个微交互博得了不少关注，主要原因在于苹果认为它不再必须，因而删除了它。到底是哪个微交互呢？就是“另存为”。

20 世纪 80 年代初，“保存”通常意味着“保存并离开”（施乐 Star），或者“保存并离开”及“保存并继续”（苹果 Lisa）。（“离开”的意思是“关闭”。）“保存并继续”最终变成了“保存”，而“保存并离开”最后就消失了。可能是因为设备内存容量更大了，能够容纳多个文档同时打开，而且处理器也不会抗议。“另存为”大约出现在 80 年代，与“保存副本为”一样，都是能让用户不必重命名即可保存一个文件的新版本。后来，有些软件同时支持“保存”“另存为”和“保存副本为”。随着时间推移，“另存为”已经深入人心，特别是在“撤销”命令得到广泛采用之后，“保存副本为”几乎绝迹。

在苹果决定去掉“另存为”之际，这个微交互的规则已经深入人心长达 30 年之久：

- 修改文件；
- 换个名字保存文件；
- 后续修改只影响新文件，之前的文件保持最后一次保存时的状态。

在 Mac OS X Lion 中，苹果好像觉得“自动保存”既然支持用户回溯到之前的版本，那么它就可以取代“另存为”了。Lion 的保存规则大致如下。

- 修改文件。
- 每 5 分钟自动保存一次改动。
- 后续修改只影响文件最新的版本。
- 通过“恢复到上一版”命令可以返回文件之前的版本。
- 可以“浏览所有版本”，这个命令触发另一个微交互：浏览版本。
- 两周之后，文件被锁定；如果不解锁或复制它，就不能再修改。

如果想另外创建一个文件，必须通过“复制”，这又是一个完全不同的微交互。

- 使用“复制”命令创建另一个文件（副本）。
- 新文件出现在当前文件旁边。

- 重命名新创建（复制）的文件。
- 后续修改影响新创建的文件；之前的文件保持最后一次（自动）保存时的状态。

新规则实际上颠倒了之前的规则：如果用户想在不同的文件中进行修改，那么他必须在修改之前就意识到这一点并做出决定。遗憾的是，这并不符合大多数人的习惯（或者更准确地说，并不符合过去 30 年来我们已经习以为常的思维方式）。这个变化彻底颠覆了既有的思维模型，取而代之的并非是更好的微交互，而是两个难以理解并且与大多数人的习惯背道而驰的微交互。大多数人在打开修改之后的文档时，根本用不着再打开之前的版本。版本控制是程序员的思维方式，而不是普通大众的思维方式。如果用户（偶尔）需要文档的早期版本，他们自己会手动打开。

一石激起千层浪。一时间，质疑、声讨声此起彼伏：“把 Pages ’09、TextEdit 都有的‘另存为……’命令去掉，我看纯属多此一举。这样完全破坏了创建新文件的一个非常普遍的工作流程，即打开一个文件，然后换个新名字保存。”怒不可遏的 Mac 博主 Pierre Igot 在他的文章“Mac OS X 10.7 (Lion): Why ditch the ‘Save As’ command?”（http://t.cn/zHg19f5）中如是说。“我真心实意地希望那是个正确的决定，可几个月过去了，事实证明它是错误的。“Web 开发者 Chris Shiflett 在他的文章“Apple botches ‘Save As’”（http://t.cn/zHg19fG）中这么说。

苹果对这件事的回应，是在 2012 年发布的操作系统 10.8 版 Mountain Lion 中悄悄地恢复了“另存为”。为什么说是“悄悄地”？因为苹果没有把它恢复到菜单里，而是将其变成了一个看不见的命令，或者说一个不可见的触发器。即使如此，仍然破镜难圆，因为这一次规则又变了。*Mac Performance Guide*（http://macperformanceguide.com/）的作者 Lloyd Chambers 在“OS X Mountain Lion: Data Loss via ‘Save As’”（http://t.cn/zW0UagS）中总结了这些变化和问题：

> 如果你修改了文章，选择“另存为”，那么**包括**编辑后的原始文档和另存的副本**在内**，都会保存修改。这就意味着，另存的不仅是一个新文件，就连原始文件也悄无声息地保存了同样的修改。假如你注意到了这个自动操作，可以手工“撤销”保存，恢复原始版本；假如你注意不到，那么将来有一天很可能会大惑不解：“这他妈怎么回事！”（想象一下客户收到货之后发现发错了。）

也就是说，在 Mountain Lion 中，“另存为”的规则大致是这样的：

- 修改文件；
- 用新名字保存文件；
- 后续修改影响新创建的文件，而对原始文件的修改也会被保存；
- 可以使用“恢复到上一版”（Revert to Last）命令恢复到之前的版本。

这意味着在“保存”和“复制”之外又增加了新规则。就这样，一个简单的、深入人心的微交互，被替换成了三个不好理解的微交互，而且没有任何解释和说明。最终，苹果在 Mountain Lion 的一次更新中，又在“保存”对话框中增加了“在原始文档中保存修改”复选框。你说有多乱吧。

通过这个案例，我们可以吸取一些教训。如果你自己都不能轻易写出或画出一个微交互的规则，用户就更难建立它的思维模型了，除非你能够给出反馈，营造一些“虚拟”的模型，让用户知道都发生了什么。其次，除非你的规则确确实实很新颖，否则用户总会带着某种心理预期去使用它。你可以违背用户的心理预期（事实上最好的微交互都是通过颠覆那些根深蒂固的心理预期才让用户感到惊喜的），前提是你的微交互必须明显要好很多，这样它对用户的价值才明显——理想情况下，应该让用户感到有立杆见影的效果。苹果其实经常能带给人们惊喜，仅举一个小例子：填写电子邮件地址时，iOS 设备的键盘上就会出现“@”符号。可是，如果你的价值反馈不能立杆见影，那么你的微交互就会让人觉得是故弄玄虚。“仅仅为了不同而不同很少

能做好，而做得更好通常才会让人觉得不同。”Dieter Rams 说。[1]

3.1 设计规则

就像每款游戏的核心是一组规则一样，每个微交互的核心同样是一组规则。游戏规则决定了游戏该怎么玩，微交互规则决定了微交互能怎么用。而创建规则的核心，就是创建一个围绕微交互如何运作的简单的非技术性心智模型。

或许设计规则时最重要的是确定目标。设计规则前，需要确定一个最简单、最明确的说法，即微交互到底是什么。最好的目标应该是容易理解（我知道为什么要做这件事），而且能够完成的（我知道我可以做到这件事）。注意，不要把目标确定为整个交互过程中的某一个步骤，目标是最终状态。比如说，登录微交互的目标不是让用户输入密码，而是让他们完成登录并进入应用程序。微交互越接近目标，成功的概率就越高。目标是规则的引擎，一切都围绕它展开（见图 3-1）。

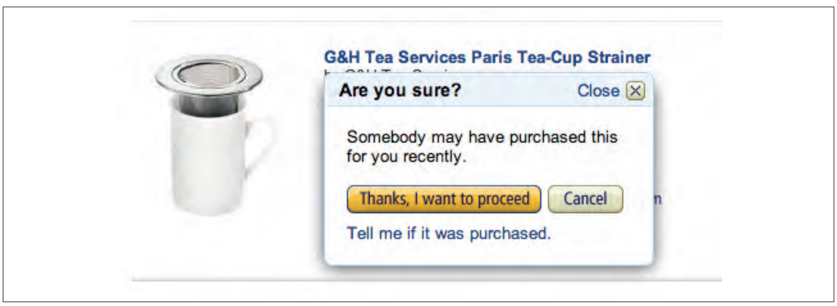


图 3-1：Amazon 上这个微交互的目标是防止用户购买别人已经为他购买的商品——防止某种情况出现，同时（某种程度上）又不影响惊喜（蒙 Artur Pokusin 和“Little Big Details”惠允使用）

注 1：据说是 1993 年说的，见 Klaus Kemp 所著的 *Dieter Rams: As Little Design as Possible*（Phaidon Press，2011）。Rams 的话很可能是受了 18 世纪德国哲学家 Georg Christoph Lichtenberg 这句话的启发：“我不知道不同会不会更好。但我知道如果更好一定会不同。”

规则的目的虽然是限制用户的动作，但不应该让用户感觉到它像规则。不应该让用户觉得需要遵循（或更坏的情况下——记住）一组严格的操作指示。你的目标是设计一个让人感觉自然而然、水到渠成的流程。换句话说，规则应该潜移默化地引导用户完成微交互的“交互”过程（见图 3-2）。



图 3-2：在苹果的 Mountain Lion OS 中，开启“听写与语音”功能，机器的风扇就会降低转速，减少噪音（蒙 Artur Pokusin 和“Little Big Details”惠允使用）

规则决定了以下事项。

- 微交互如何响应被激活的触发器。用户单击图标时会发生什么？（参见本章 3.1.5 节“不要从零开始”。）
- 交互期间用户可以进行什么操作。用户可以取消下载、调节音量，或者手工触发通常是自动触发的操作（比如检查新邮件）吗？
- 动作发生的顺序及时间。比如，在“搜索”按钮变成活动状态前，用户必须在搜索框中输入文本。

- 使用什么数据以及数据的来源。微交互是否依赖地理定位、天气状态、当前时间、股票价格？如果是，这些数据从哪里得来？
- 简洁的相关配置及参数。规则从整体上来看是可以用算法表达的，而微交互的某些部分又是由算法驱动的。（参见本章 3.4 节"算法"。）
- 什么时候提供什么反馈。规则可以表明哪个"步骤"应该有反馈，以及后台都在进行什么操作。
- 微交互处于什么模式（mode）。这里所谓的"模式"是指规则中应该尽可能避免的"分叉"。但有时候这种"分叉"又是必要的。比如在很多天气应用中，输入城市就是一个输入模式，它独立于默认的查看天气模式。关于模式的更多内容，请参见第 5 章。
- 微交互是否重复以及多久重复一次。微交互是一次性的活动，还是个循环往复的过程？关于循环的更多内容，请参见第 5 章。
- 微交互结束时会发生什么。微交互结束时是切换到另一个微交互，还是自动消失？亦或永远不会结束？

用户可能知道也可能不知道规则的存在，而规则让自己可见的方式有两种：允许做什么和不允许做什么（见图 3-3）。这两种方式都反馈的时机（参见第 4 章）；当然，正如第 1 章开头那个故事所揭示的，有时候用户的心智模型与规则生成的概念模型不一定匹配。

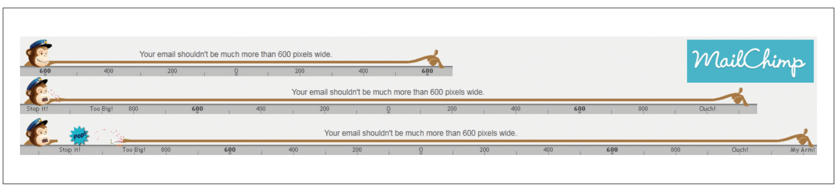


图 3-3：MailChimp 为了告诉用户不能做什么，让可怜的黑猩猩伸出胳膊来表示宽度，一旦超过指定宽度，黑猩猩的胳膊就会掉下来（蒙"Little Big Details"惠允使用）

我们还是以开灯这个最简单的微交互为例。开灯的规则如下：

- 如果开关被按下，灯点亮并保持；
- 如果开关再次被按下，灯熄灭。

非常简单。[2] 假如我们在开关上增加一个运动传感器，规则就会复杂很多。

- 每 3 秒钟检查一次运动。
- 如果有物体移动，确定是否与人体大小相仿？（一只猫跑过来，不必开灯。）
- 如果是，开灯。
- 每 3 秒钟检查一次运动。
- 有物体移动吗？
- 如果没有，等 10 秒钟，关灯。

当然，这些规则未必无懈可击。3 秒检测一次是不是太长？还是说太频繁了，这么频繁是不是太耗电了？或许你真想让猫跑进屋的时候也能自动开灯。我相信，读者中一定有不少人有过在洗手间因为传感器没有检测到运动而自动关灯的体验，这种情况下 10 秒钟也许太短了。显然，规则决定什么事情发生，以什么顺序发生，进而影响用户体验。

3.1.1　生成规则

设计规则最简单的方式，就是一开始先把你能想到的规则大体上记录下来。一般来说，能够记录下来的都是微交互期间的主要动作，只要把它们一步一步地按照顺序记录下来即可。比如把一件商品放到购物车里，一开始规则可能如下所示：

1. 在商品页面上，用户单击“添加到购物车”按钮；
2. 商品被添加到“购物车”中。

上述规则很简单。但随着设计的细化，以上规则会变得复杂起来。

注 2：当然，真实的电灯开关的工作原理不是这样的。按下开关就会接通电路，电流流过灯泡。再按下开关就会切断电路。但用户不必知道这些物理上的细节，只要理解规则即可。

1. 在商品页面上，检查用户以前是否购买过这个商品。如果购买过，把按钮的标签由“添加到购物车”改为“再次添加到购物车”。
2. 检查用户的购物车中是否已经有了这个商品。如果有了，把“添加到购物车”改为“再添加一个到购物车”。
3. 用户单击按钮。
4. 把商品添加到购物车。

差不多如此。而这还只是设计一个如图 3-4 所示按钮的规则。进行真正设计时，规则可能还会更多。

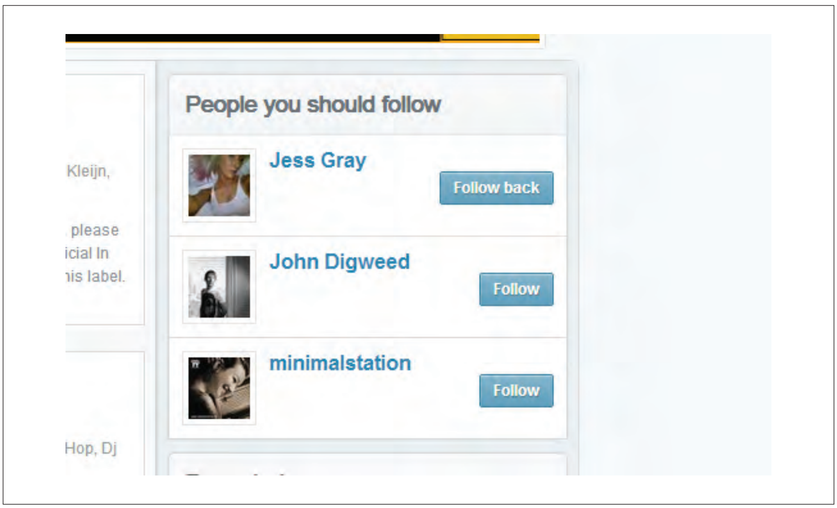


图 3-4：简单按钮的规则。如果有人在 Mixcloud 上已经关注了你，“关注”按钮会变成“互相关注”（蒙 Courtesy Murat Mutlu 和“Little Big Details”惠允使用）

当然，如果能够可视化地呈现规则，那就更好了。有时候，逻辑关系图非常有用（见图 3-5）。

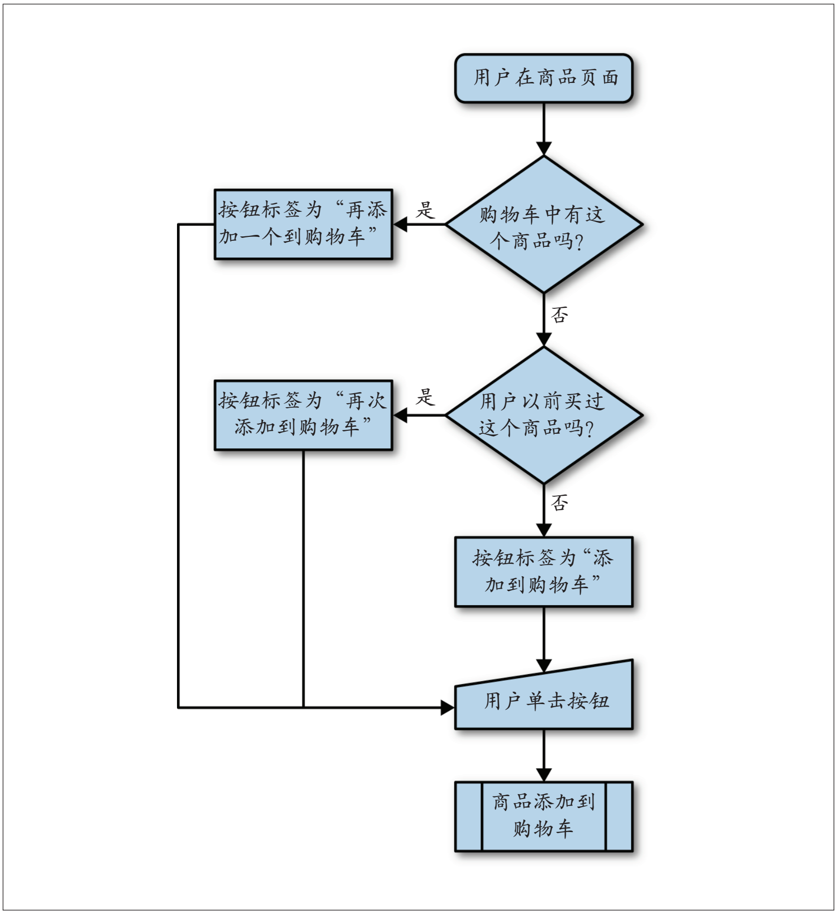


图 3-5：表示规则的逻辑关系图

规则关系图能够让人对规则之间的逻辑关系一目了然，知道哪些动作变得（过度）复杂了。另外，关系图也能帮人发现一些在纯文本形式下发现不了的错误。图 3-6 展示了一个细微的规则。

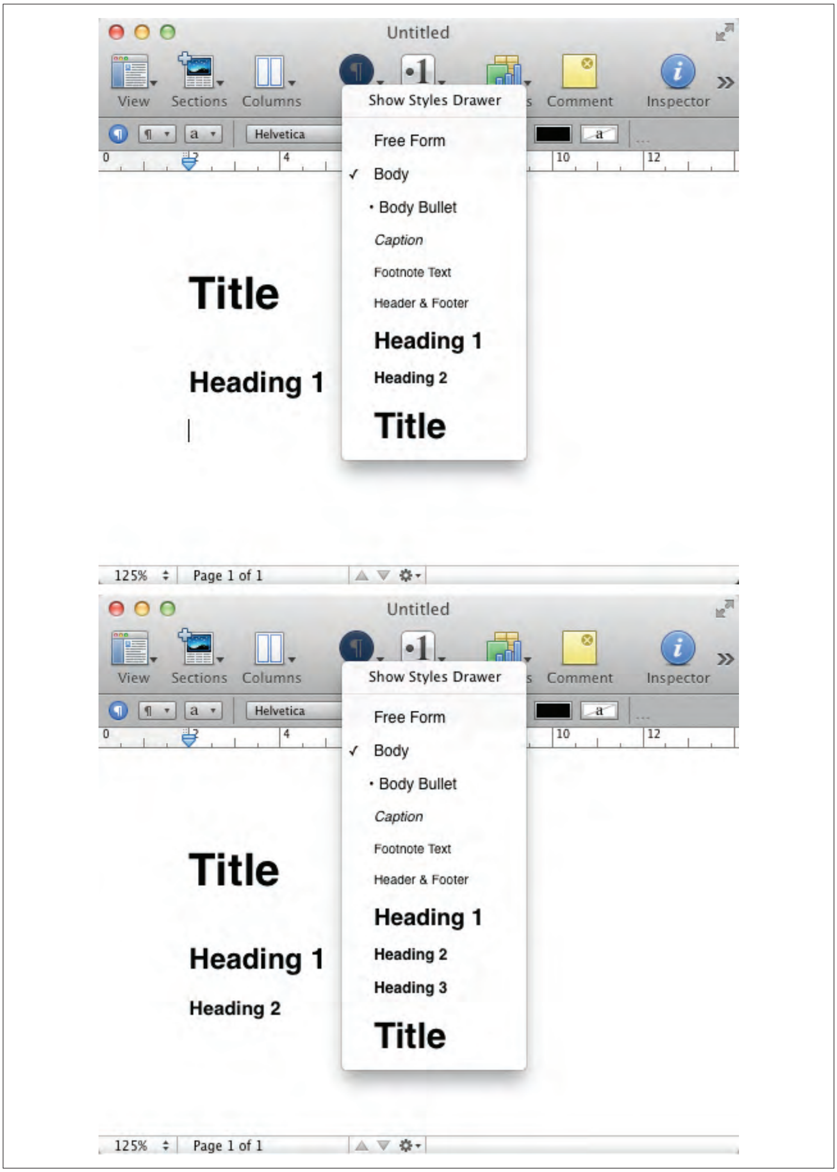


图 3-6：苹果的 Pages 会自动添加更低级别的标题样式，但前提必须是已经使用了当前最低级别的样式。比如，只有在已经使用了 Heading 2 的情况下，才会显示 Heading 3（蒙“Little Big Details”惠允使用）

3.1.2　动词与名词

可以把微交互想象成一句话。这句话中的动词代表用户要采取的动作，名词代表辅助实现该动作的对象（见图 3-7）。比如，通过滑动条可以调节音量大小。动词就是用户可以做什么（调节音量大小），而名词就是通过什么来调节（滑动条）。



图 3-7：如果有朋友“喜欢”你在 Facebook 上的跑步记录，你就能在使用 Nike+ 应用时从耳机中听到加油的欢呼声（蒙“Little Big Details”惠允使用）

微交互中的所有对象（包括 UI 界面中每一部分、表单中的每个元素和控件、每个 LED 灯），都是带有特征和状态的名词。而规则定义了这些特征和状态（见图 3-8）。以一个简单的下拉菜单为例，它有两个状态：打开和关闭。打开时，菜单选项会显示出来，这些选项就是菜单的特征。除了选项，菜单还可以有其他特征，比如最多显示的选项个数和每个选项标签的最大长度。菜单还可以有其他状态，比如悬停打开和悬停于选项上时显示工具提示条。诸如此类的细节都要由规则来定义。（动词也有特征。比如，某个任务完成得多快？一个动作花多长时间？这些同样也要由规则来定义。）

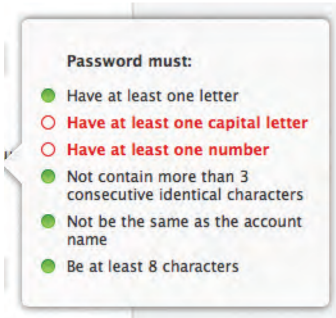


图 3-8：修改 Apple ID 的密码时，程序会随着用户输入逐个核对“必须……”条件。这种直观的方式揭示出了微交互的约束作用（蒙 Stephen Lewis 和“Little Big Details”惠允使用）

微交互中的每个名词都应该独一无二。如果有两个相同的名词，要想办法把它们合二为一。还要注意，两个（或多个）看起来相同的名词，它们的行为也应该相同（见图 3-9）。两个看起来类似的按钮，它们的行为不能差得太远。行为不同的对象看起来应该完全不一样。类似地，相同的名词在不同的地方也不能行为迥异。Android 系统中的“后退”按钮就是一个著名的例子，它能把用户带到什么地方？在不同的应用中答案也不同：有时候是前一个面板，有时候则是完全不同的应用（参见 Ron Amadeo 的文章“Stock Android Isn’t Perfect”：http://t.cn/zlzCee4）。

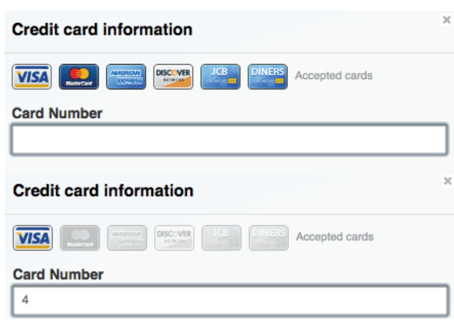


图 3-9：GitHub 不让用户选择信用卡，而是根据用户输入的卡号自动判断并作出选择（蒙“Little Big Details”惠允使用）

微交互的最高境界通常都是给用户提供多种多样的动词，但只提供最少的名词。

3.1.3 屏幕与状态

实现规则时很有诱惑力的一种做法，就是为规则中的每一步都设计出一个屏幕；换句话说，就是为整个微交互设计一连串安装向导式的UI。这种做法适合一些特殊的微交互。那怎么个特殊法呢？每个步骤要明确、不连续，不会反复出现，甚至只要过一遍就完事。但是，对于大多数微交互而言，这种做法会破坏整个工作流程，造成不必要的割裂。事实上，采用独立屏幕往往不如利用状态变化。因为状态变化可以在不加载新屏幕的情况下，渐进地揭示每时每刻必须做出的决定或操作的控件（见图 3-10）。

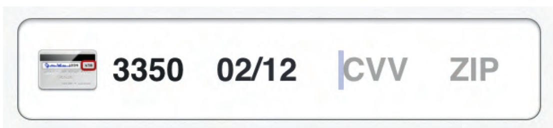


图 3-10：在 iOS 应用 Square 中输入 CVV（Card Verification Value，信用卡安全码）时，信用卡图片会翻到另一面，让用户一眼就知道到哪里找到该号码（蒙 Dion Almaer 惠允使用）

随着用户一步一步地遵循规则进入微交互，微交互中的对象（名词）可以（或者说应该）改变，以反映每时每刻的变化。每次改变都是一种应该在设计时考虑的状态（见图 3-11）。

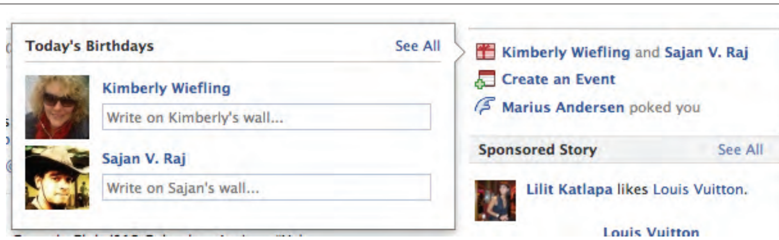


图 3-11：如果几个朋友的生日恰好在同一天，Facebook 的生日微交互支持你同时给他们留言（蒙 Marina Janeiko 和“Little Big Details”惠允使用）

用户可以与之交互的任何对象都（至少）有三种状态。

- 邀请/默认状态
  第一次呈现在用户眼前时的状态。这个状态也适合呈现一些预先加载的数据。

- 活动状态
  用户与之交互时，对象有什么反应？

- 更新后的状态
  用户停止与之交互后，对象有什么变化？

我们以简单的拖放为例。可以拖放的对象，它的初始 / 默认状态应该看起来就是可以拖动的。而且，除此之外，对象本身（以及光标）应该有一个悬停状态，表示对象可以拖动。而在被拖动期间，对象还可以变成另一种状态。（此时，屏幕 [另一个名词] 也可以变成另一种状态，表示可以把对象放在哪里。）最后，还有一个放置状态，有可能就是恢复为默认状态（见图 3-12）。

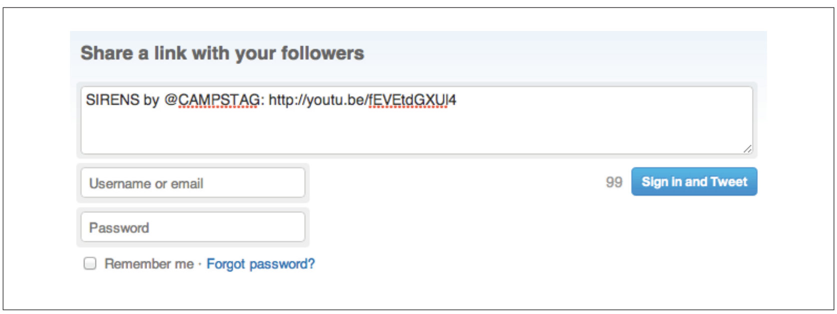


图 3-12：在 Twitter 上，分享链接的按钮有两个空闲状态：登录和未登录。如果未登录，单击该按钮可以同时登录和分享（蒙 Rich Dooley 和 “Little Big Details” 惠允使用）

微交互的设计师应该关注每一个状态，因为每一个状态都需要向用户传达信息，告诉用户当时发生了什么，即使什么也没有发生。

3.1.4　约束条件

设计规则时需要考虑业务性、环境性及技术性的约束（见图 3-13），包括但不限于以下几个方面。

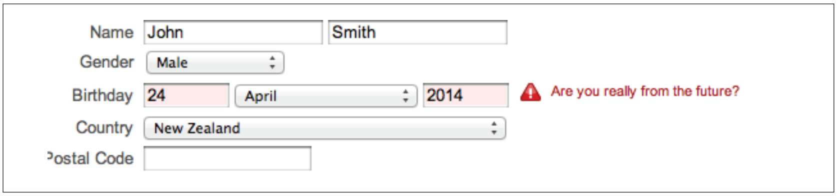


图 3-13：Yahoo! 的注册微交互不允许填写将来的日期，因此只提供下拉列表让用户从中选择，从而完全避免错误（蒙“Little Big Details”惠允使用）

- 可用的输入和输出方法。有键盘吗？有扬声器吗？
- 输入内容的类型和范围。比如，密码框允许输入几个字符，或者用户可以把音量调到多大。
- 要付出哪些代价。不仅仅从金钱角度（例如，如图 3-14 所示的那样使用某些数据服务），还要从资源消耗或占用角度考虑要付出什么代价。每 10 秒向服务器发送一次请求可能会给服务器带来极大负担，同时也会导致设备电池消耗过快。

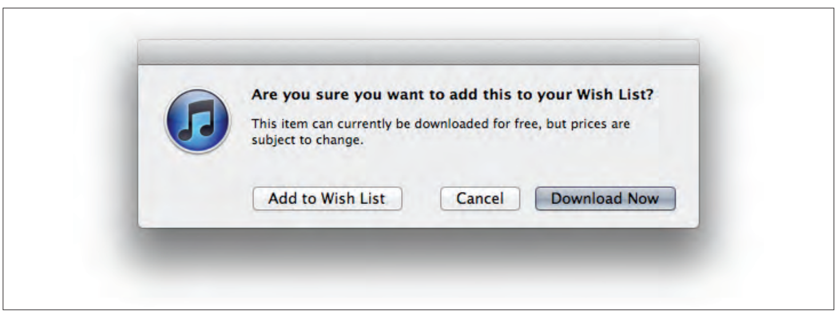


图 3-14：在向愿望清单里添加免费商品时，iTunes 会告诉你只要下载就可以免费安装了（蒙“Little Big Details”惠允使用）

- 可用的数据有哪些。可以通过传感器收集到哪些数据？有哪些服务或 API 可以用来取得位置、新闻、天气、时间等信息？

- 可以收集哪些数据。可以收集和使用用户个人的哪些(行为)数据?

最后两个约束条件可以让你不必从零开始。

3.1.5　不要从零开始

对任何微交互而言，触发器被启动后的第一个问题都是：我对用户以及用户所处的情境了解多少？你肯定了解一些情况，而有些情况可以用于改进微交互（见图 3-15）。

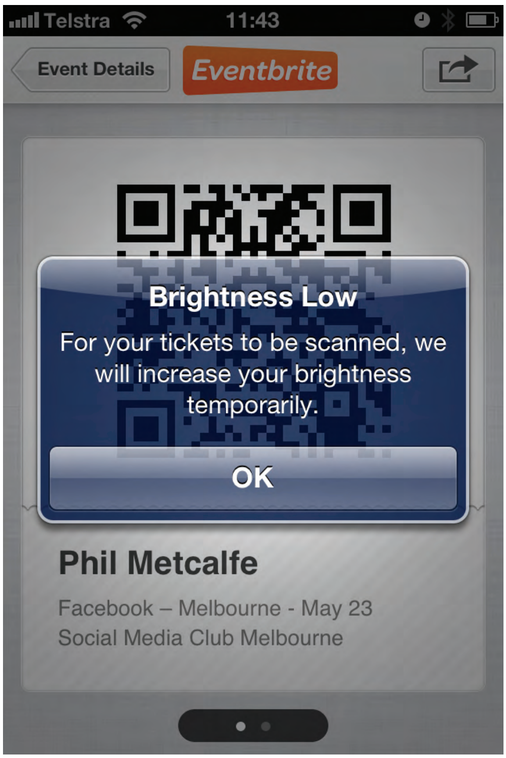


图 3-15：iOS 应用 Eventbrite 为了方便扫描二维码，会临时增加 Mobile Ticket 屏幕的亮度。对于这个情境而言增加亮度是有必要的，不过这个警告框似乎没什么必要（蒙 Phil Metcalfe 和 “Little Big Details” 惠允使用）

以下是一些常见的例子：

- 用户使用的平台 / 设备；
- 当天的时间；
- 房间的噪音；
- 微交互的持续使用时间；
- 用户是不是在开会；
- 用户是不是一个人；
- 电池还剩多少电量；
- 设备的位置和方向；
- 用户刚才做过什么。

有些并非直接来源于用户的数据可能会更有用（见图 3-16）。



图 3-16：Google+ 会根据你朋友的职业猜测你在哪里上班（蒙 Artem Gassan 和“Little Big Details”惠允使用）

最后一项、可能也是最重要的一方面数据有赖于收集用户的行为信息。记住，“从系统资源角度说可能是这个问题”的观点早就过时了。今天，即使是小功率的家用电器都有了足够的内存和处理器去收集这些信息。因此真正的问题在于，我们能否说服人力资源（开发人员）相信这是一件值得做的事。（事实上，这是绝对值得的。）当然，设计师应该明白：必须尊重隐私。假如微交互涉及一些敏感话题（比如体检结果），那你很可能涉嫌收集个人隐私。怎么区分呢？可以向自己提问：微交互收集到的信息是否会让用户感到尴尬、羞愧或威胁？如果是，就不要收集了。用户宁要没有个性的体验，也不想暴露隐私（见图 3-17）。

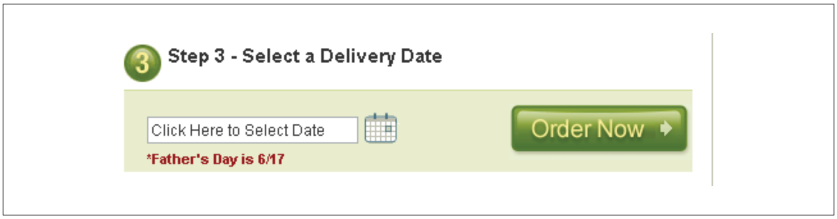


图 3-17：Pro Flowers 在你选择送货时间时会向你显示最近的节目（蒙 Gabriel Henrique 和“Little Big Details”惠允使用）

这些数据经常可以混合使用，比如：每天 10 点，用户做某某事，因此如果用户恰好这个时间触发微交互，就为他提供某某。比如：用户每隔一定时间到某个地方，就会做某某事。再比如：用户每次通过移动设备登录，都只会阅读。图 3-18 展示了这么一个例子。

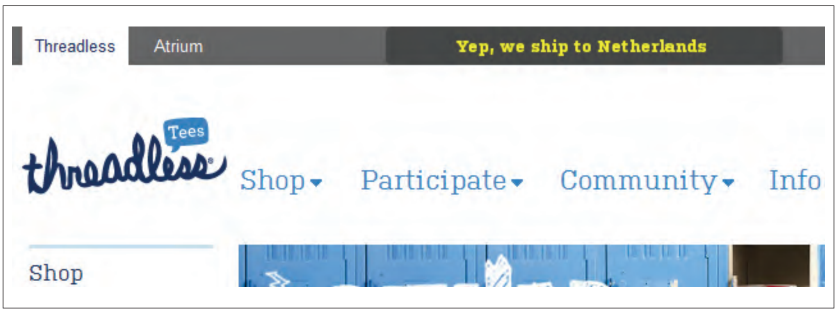


图 3-18：第一次登录时，Threadless 会让你知道能否把货送到你所在的国家（蒙“Little Big Details”惠允使用）

总之，就是要善于利用情境和之前的行为预测或增强微交互（见图 3-19）。可以把收集用户数据的过程当成持续的用户研究，在此基础上进行分析就有可能了解用户如何使用微交互，从而做出有针对性的调整。比如，通过收集用户行为数据，你会发现应该让高级用户利用不可见的触发器跳过规则的某些步骤。导航应用 Waze 让高级用户滑动（而不是按下）按钮就可以直接打开导航，省了两次点击操作。

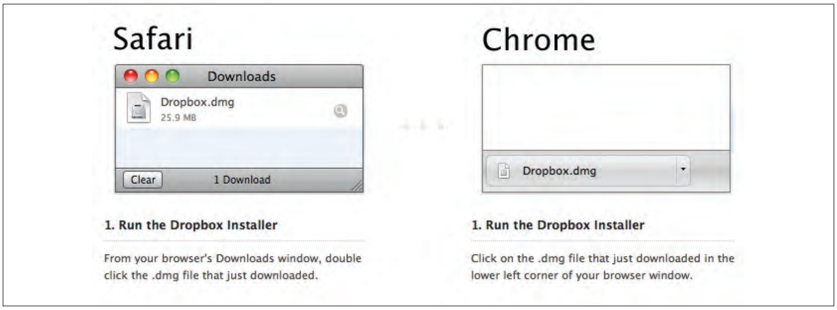


图 3-19：Dropbox 根据用户使用的浏览器改变下载说明（蒙 Mikko Leino 和“Little Big Details”惠允使用）

3.1.6 理解复杂性

我们在第 1 章介绍过的发明剪贴操作的 Larry Tesler 曾经有一个观点，非常适合设计规则：特斯勒复杂性守恒定律。这个定律的大意就是，所有活动都有内在的复杂性，超过了某个临界点，简化是不可能的。既然如此，唯一的问题就是如何对待复杂性。要么让系统来处理，不让用户介入；要么让用户来处理，把更多的决定权（包括控制权）交给用户（见图 3-20）。



图 3-20：即使是难用的 iCal，在选择时间的微交互中也有一个值得称道的规则。这个微交互不让你自己计算活动需要多长时间，而是会在你选择结束时间时自动给出活动的持续时间。这是微文案的一个实际应用（蒙 Jack Moffett 惠允使用）

设计微交互时，人们大都倾向于减少用户控制，而让微交互处理大多数复杂性。对此，我们想给出一个忠告：有些微交互必须由用户来控制，但即使如此，系统仍然需要处理一定的复杂性（见图 3-21）。

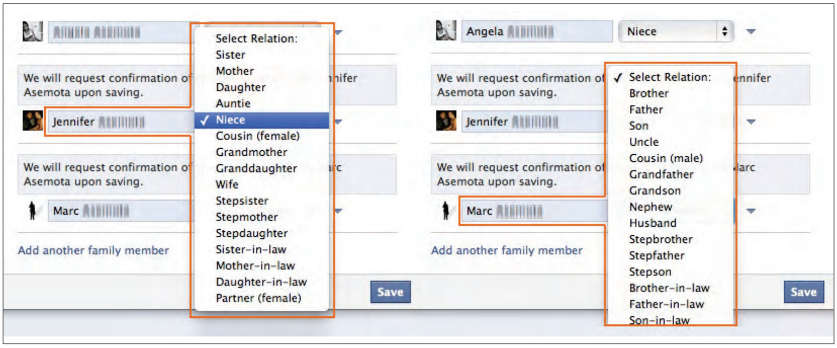


图 3-21：在 Facebook 上添加新家庭成员时，Facebook 会自动判断被选家庭成员的性别，并尽可能调整列表框中的成员关系（蒙 Stefan Asemota 和 “Little Big Details” 惠允使用）

首先，要找出最核心的复杂性在什么地方，确定用户掌握着哪一部分以及何时需要介入。如果用户介入绝对必要，则适时将控制权交给用户（见图 3-22）。

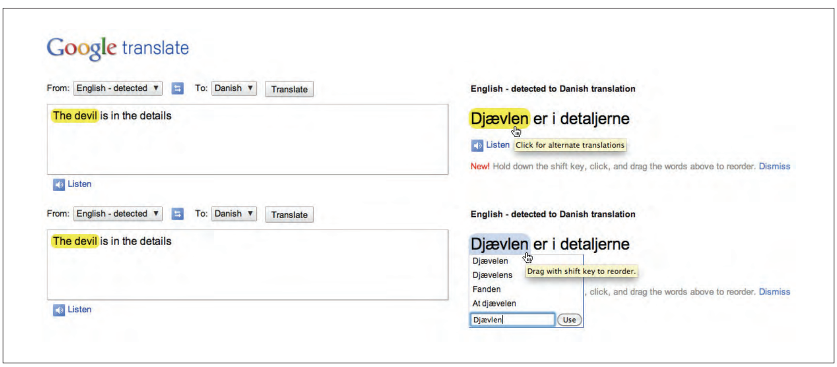


图 3-22：光标悬停在 Google Translate 中的译文上时，对应的原文会自动高亮。可以更改译文，但必须与原文一一对应（蒙 Shruti Ramiah 和 “Little Big Details” 惠允使用）

从处理复杂性的角度来说，计算机肯定要比人更能胜任。如果你的微交互中涉及下列复杂性，那就让系统处理：

- 快速计算；
- 同时执行多个任务；
- 记忆很多东西；
- 检测复杂模式；
- 从大数据集中搜索特定的物品。

当然，除了让系统承担起处理复杂性的任务外，你还必须足够聪明，知道给用户提供哪些选择以及它们的默认值。

3.2 有限的选项和聪明的默认项

提供给用户的选项越多，微交互的规则也就越多。一般来说，规则越少则微交互就越容易理解。因此，要尽量少给用户选择，同时提供聪明的默认项。

对于微交互来说，最好能提示（或自动执行）用户最可能采取的下一个动作。提示的方式可以是去掉其他所有选项，或者视觉上给出变化（比如，提供一个大按钮）。正如游戏设计师 Jesse Schell 在他的书 *The Art of Game Design*（CRC Press）中所说的："控制了用户的眼睛，就控制了用户的脚步。"

知道了用户最可能采取的下一个步骤，你就可以自动地、在没有用户干预的情况下执行或呈现该步骤（见图 3-23 和图 3-24）。这是把微交互连接为一个整体的好办法（参见第 6 章的 6.5 节"编排微交互"）。

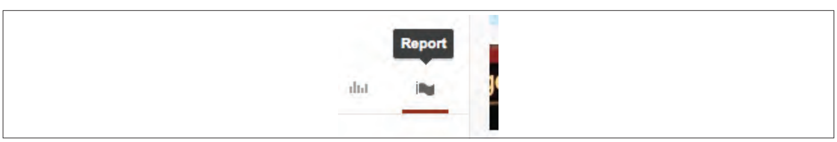


图3-23：在YouTube上按Report（举报）按钮会自动停止播放你想举报的视频，它会为你执行下一个可能的动作（蒙 Aaron Laibson 和"Little Big Details"惠允使用）

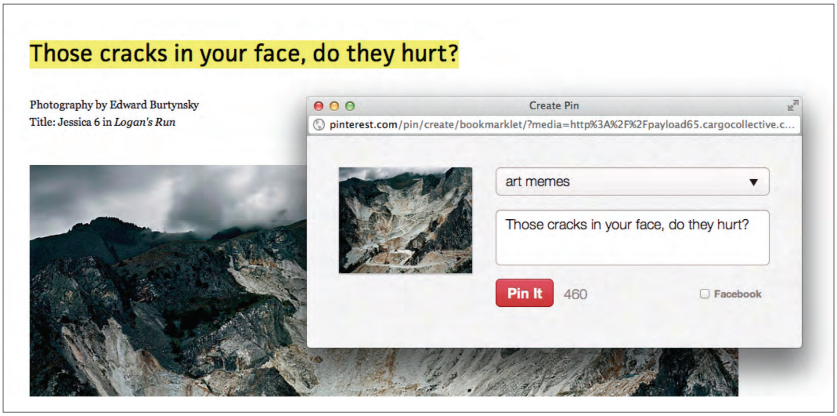


图 3-24：在把图片添加到图片分享社交网站 Pinterest 时，页面中选中的文字会自动填充为图片的标题（蒙 Louisa Fosco 和 “Little Big Details” 惠允使用）

用户手里的每个选项都至少对应于一条规则，因此减少规则的最好办法就是给用户提供有限的选项。简言之，选项越少越好。微交互致力于把一件事做好，理想情况下用户看不到选项，一以贯之的只有聪明的默认项。所有人都只要操作一下，然后一劳永逸：规则一结束后是规则二，然后是规则三。这正是谷歌的搜索框成为 21 世纪早期最有效率（至少在互联网上被使用最多）的微交互的原因。每个人都遵循相同的规则：

- 输入文本按下（大大的）搜索按钮；
- 看到搜索结果。

当然，谷歌也免不了多给了用户一个选项：“手气不错”按钮。点这个按钮会直接显示排名最高的搜索结果。“手气不错”按钮只有 1% 的用户使用，而且据报道称，这个按钮每年都会让谷歌损失 1 亿美元的广告收入。2010 年，谷歌在推出 “Google Instant”（瞬时搜索）时果断删除了 “手气不错”。Google Instant 能随着用户输入实时显示搜索结果，因此根本没有按 “手气不错” 的必要。[3] 于是，规则变成了这样：

注 3：Nicholas Carlson, “Google Just Killed The ‘I’m Feeling Lucky Button’”（谷歌扼杀了“手气不错”按钮）*Business Insider*，2010 年 9 月 8 日。

- 输入文本；
- 显示搜索结果。

这个规则已经没办法再简化了，除非谷歌能猜出来你想搜索什么，然后立即把结果显示给你。

对微交互而言，两个选项可能都太多了。但这并不是说不能提供选项，比如温度设定（热、温暖、冷），而是说提供更多选项造成规则骤变是不明智的。规则的这种变化很可能导致微交互进入另一种模式（参见第 5 章）。最常见的例子就是“忘记密码了？”链接，很多登录微交互都提供这个选项。单击该链接，用户就会进入另一种模式，而该模式最终可以把用户带回主模式输入想起来的密码。

要替用户做出某些默认选择，某些情况下应该给出与决定相关的说明。比如苹果的日历提醒功能，在显示提醒（如“15 分钟后开会”）时，还会显示一个“再睡一会”按钮。可是，并没有同时显示“再睡”多长时间（我自己的经验是 15 分钟，有点太长了），也没有给出修改这个默认项的选项。实际上，把“再睡 15 分钟”作为按钮的标签应该更好，因为它说明了规则。

最明显的默认项应该是大多数人大多数情况下会用到的。即使你觉得这个选项不应该替用户选择，也应该让它看起来非常明显。最常见的例子就是“确定 / 取消”按钮。“取消”被按下的机会一般比“确定”少得多，因此“确定”应该更容易看到（要么大一些，要么颜色上加以区分）。另外，不要忘了“回车”键（当然，前提是用户设备上有这个键）。按下“回车”应该执行默认操作。

如果必须向用户提供一个选项，那么要记住：怎么呈现这个选项会影响用户的选择。位于列表顶部或底部的项目，比位于列表中部的项目更容易被记住。而高亮的项目被选择的机会也比未高亮的项目多。如果需要用户做出一连串决定，那么先从简单的开始，慢慢地给出复杂的选项。美国美林银行负责可用性的副总裁 Colleen Roller 说过：“在理解可用选项并能够对它们逐个比较和评估的条件下，人们最有信心

做出决定。选项越少，区分越容易，评估越简单。”[4]

既然每个选项（至少）对应一条规则（而且，别忘了我们希望规则越少越好），那么你提供给用户的选项就该非常有意义。有意义的选项可以影响用户完成微交互目标的方式，甚至会影响目标本身。怎么样才算有意义？比如登录 Facebook，或者说输入用户名和密码。什么是没有意义的选项呢？就是你无论选什么，都不会影响结果。亚马逊的 Kindle 应用要求用户选择高亮文本的颜色，但用户却不能搜索也不能导出高亮的文本。这种选项就没多大意思了，还不如通过默认的高亮微交互来自动设置。是否有意义，有什么判断标准吗？只要问一个问题：提供这个选项能让用户感觉有意思、有价值，或者心情舒畅吗？如果答案是不能，那还是算了吧。

去掉选项有一个好处，即避免很多极端情况出现。极端情况指的是那些非常难解决的问题，而且只是偶尔才可能碰上，只适合小众（高端）用户。极端情况可能会导致微交互跑偏（warp），让你的设计偏向不常见的场景，而不是最常见的场景。极端情况是微交互的天敌，必须尽一切可能避免，包括修改规则，消除其产生的基础。举个例子，如果“出生年份”是一个文本框，那么用户很可能在里面输入无效日期，比如写成了未来的某一年。把文本框改成只能选择的下拉菜单，就可以消除这种极端情况。

3.2.1 控件和用户输入

大多数微交互都会让用户输入信息。为此，必须决定要使用什么控件，如何展示这些控件。以调节音量这个最简单的微交互为例，音量调节可以分为三个状态：调大、调小和静音。为此，可能的方案包括三个按钮、一个滑动条、一个拨号盘、两个按钮、一个滚轮、一个滑动条和一个按钮，等等。

注 4：“Abundance of Choice and Its Effect on Decision Making”（选择多样性及其对决策的影响）*UX Matters*，2010 年 12 月 6 日。

控件的选择需要权衡易操作和易辨识这两个方面（见图 3-25）。易操作意味着每个指令对应一个控件。对音量调节来说，就是有三个按钮：一个用于调大音量、一个用于调小音量，还有一个用于关闭声音。易辨识意味着让一个控件承载多个操作。对音量调节来说，就是要选择滑动条或滚轮。

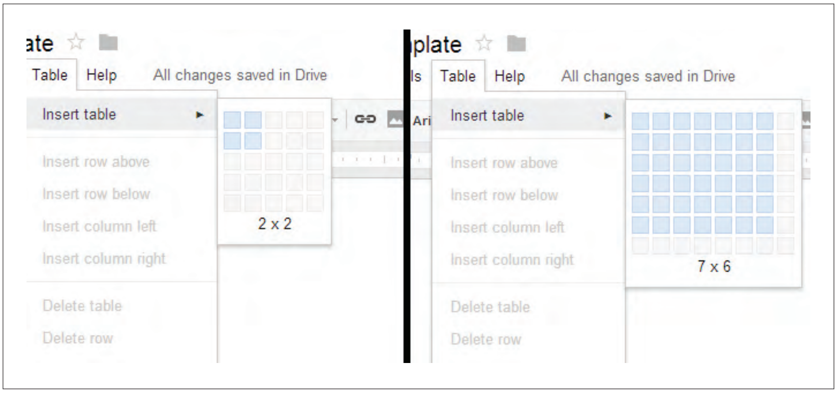


图 3-25：谷歌 Drive（云端硬盘）的插入表格微交互提供了悬停时扩展的窗口，用户可以在其中直观地选择表格大小（蒙 Kjetil Holmefjord 和“Little Big Details”惠允使用）

对于需要不断重复的微交互，一般应该让它易辨识，除非是那种可以迅速完成且不可能出错的动作（比如按下电话上的静音按钮马上可以静音，而如果给按钮加入调小音量功能，那后果恐怕是灾难性的）。对于只要做一次或者偶尔才需要做的微交互，应该让它易操作，把所有选项都显示出来，让人一目了然，不要求用户有什么先行的经验。

文本字段应该接受填入其中的任何内容，并假设这些内容可能来自任何地方，特别是用户内存中的剪贴板。比如，表单中的电话号码字段应该支持用户输入下列任何形式的内容：(415) 555-1212、4155551212、415-555-1212。文本字段特别需要系统设计师所说的“必要的多样性”（requisite variety），即适用各种条件的能力。这意味着通常要通过代码“修正”输入，使其达到代码或数据库的要求（图 3-26 是一个反面的例子，图 3-27 是一个正面的例子）。

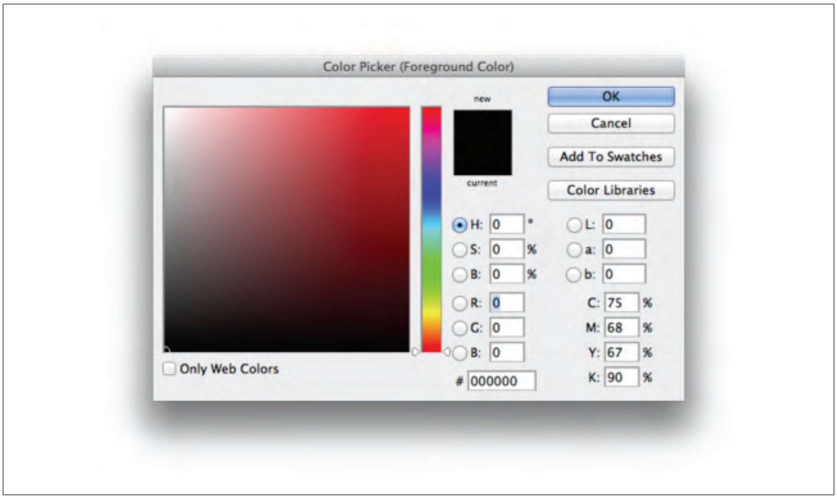


图 3-26：Adobe Photoshop 的拾色器微交互提供了一个输入十六进制颜色值的文本框。不过，这个文本框不够聪明，不会去掉用户贴到里面的 # 符号（蒙 Jack Moffett 惠允使用）

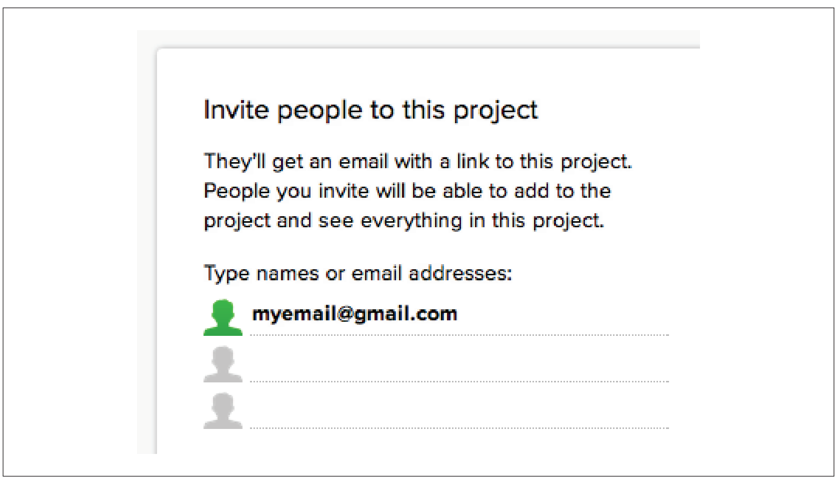


图 3-27：37signals 的 Basecamp 做得很好。在贴进一个类似 Jane Smith<myemail@gmail.com> 这样的电子邮件地址时，它会自动去掉多余的字符，只留下电子邮件地址（蒙 Harpal Singe 和“Little Big Details”惠允使用）

对于下拉菜单之类的列表中的各项顺序，应该慎重对待。有时候，可以按照字母顺序或最后使用等既定的习惯排序。但有时候，看起来似乎不合逻辑的排序方式或许更有意义。举个例子，如果用户大多来自美国，那么让他们向下滚过 20 个字母，找到 U 开头的这个国家就不合适了。这时候，应该把美国放在列表开头或者作为默认值，尽管有点违背常理。

有时候，多提供一些控制手段是合理的，特别是在微交互会被用户频繁用到的情况下。比如，设计一些快捷方式。在桌面软件中，惯例都是提供快捷键，像 Command-Q 表示退出，而在触摸屏或轨迹球设备中，可以是手势（一般是多点触摸的手势）。前提是不要让快捷方式干扰了流程的主要操作。任何重要的操作都应该有一个显眼的控件，让用户手动可以操作。

3.2.2 预防错误

设计规则时的一个任务就是预防错误（见图 3-28 和图 3-29）。微交互应该遵循 20 世纪 60 年代丰田汽车公司传奇工程师新乡重夫发明的防呆原理（Poka-Yoke Principle）。防呆（防错）原理是说，在设计产品和流程时，就要考虑根本不让用户出错。比如苹果的 Lightning 插头，正反都可以插入 iPhone 或 iPad 上的接口，而之前的 30 针插头（以及所有 USB 插头）则需要先找到方向才能插进去。

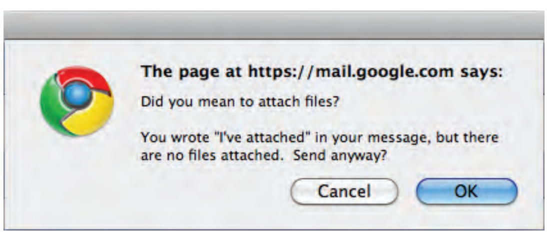


图 3-28：Gmail 在你发送邮件之前会提示你是否忘了添加附件（蒙“Little Big Details”惠允使用）

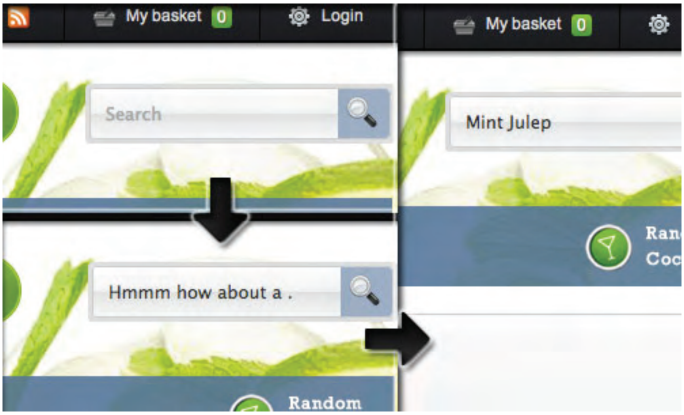


图 3-29：在 Make Me a Cocktail 中，如果不输入任何搜索关键词就点搜索按钮，它不会显示错误消息或者不做反应，而是会随机显示一种鸡尾酒（蒙 Nick Wilkins 和“Little Big Details”惠允使用）

类似地，设计微交互时也应该不允许用户犯错（见图 3-30）。或许这意味着减少用户的控制和输入，但减少微交互中用户的选择通常都是正确的。

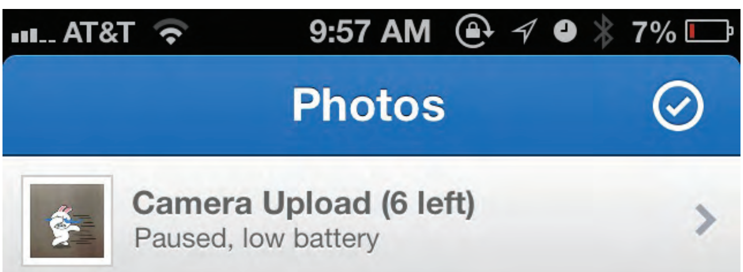


图 3-30：iOS 版 Dropbox 会在电量低的情况下暂停上传（蒙“Little Big Details”惠允使用）

理想情况下，你的微交互应该在用户没做错任何事（因为不应该有用户会做错的事）的情况下，一条错误消息都不出现，而只在系统本身

不能正确响应的情况下才会出现错误消息。弹出错误警告是懒人的做法。如果确实发生了错误，那微交互应该首先想办法确定怎样解决问题（见图 3-31）。

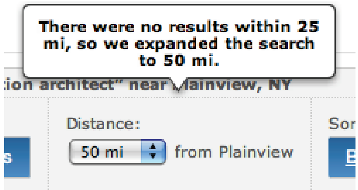


图 3-31：Meetup.com 会在没有搜索结果的情况下调整搜索条件（蒙 Michael J. Morgan 和“Little Big Details”惠允使用）

同样地，根据这个原理也可以防止用户以不适当的方式使用微交互（见图 3-32 和图 3-33）。比如，你可以禁止在评论中发表污言秽语。



图 3-32：What do you love? 不让你输入污言秽语，它会把不雅的内容转换成“kittens”并显示相应结果（蒙 Zachary Reese 惠允使用）



图 3-33：Twitter 不允许你重复发表相同的推文，主要是为了保证系统不被滥用。虽然可能会多占用些系统资源，但最好还是在用户按“发推”之前检查内容是否重复（蒙 Sindre Sorhus 和“Little Big Details”惠允使用）

3.3　微文案

微文案，即标签、说明及其他文本片段，是理解规则必不可少的一部分。微文案属于那种固定不动的反馈或前馈。一个微文案本身就可能是一个完整的微交互，比如 Facebook 的“Like”按钮，就完全构建于蓝色的 Like 文本之上。

系统触发器可能会导致一个非常重要的微文案出现，而此时这个微文案的意义重大。比如，在一家商店的“联系我们”的告示上，“抱歉，我们已经打烊了”旁边可以显示工作时间之外的联系电话。这个电话号码就是一个完整的微交互。

几乎所有微交互都应该首先保证自己提供的文本是必要的，但说明性的文案则不一定。一般来说，用不着在登录表单上方来一句“请登录”，用户也能理解自己应该干什么。如果需要显示文本，一定要尽可能简短。正如温斯顿·丘吉尔所说：“越短越好，大白话更好。”

如果标签就足够了，那就绝不要使用说明性文案。如果有一个“下一步”或“继续”按钮，那么“按‘下一步’继续”就是不必要的。如果为了省空间必须截短标签（比如相册的名字），那么应该提供一种手段让用户能够看到全称，比如悬停或翻转（桌面 /Web 应用）、触摸 / 点击（移动应用）。有时候，特别是在物理按钮上，由于空间不够，

制造商经常会只印出一个词的几个字母，看起来就像自编汽车号牌一样难以理解。这样不好，如果放不下一个词，不如放个图标。

避免使用可能误导人的标签。比如，图片共享服务器 Flickr 用于导航的两个链接分别是“← 前一张”和“后一张 →”。可是点“前一张”打开的却是后来才发的新图片，而“后一张”打开的则是更早的一张老图片（见图 3-34）。

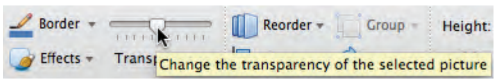


图 3-34：微软 PowerPoint 功能区中调整透明度的滑动条。没有标签说明你在调高还是在调低透明度，而且变化直到释放鼠标后才会发生（蒙 Jack Moffett 惠允使用）

放置标签的最佳位置是操作地点上方，其次是被操作的对象之上或之中。可以参考 Luke Wroblewski 的两篇博文“Top, Right, or Left-Aligned Form Labels”（http://www.lukew.com/ff/entry.asp?504） 和“Web Form Design: Labels Within Inputs”（http:// www.lukew.com/ff/entry.asp?687）。原因很简单，这样让人一眼就能同时看到标签和对象。换句话说，眼睛不需要在两个对象间移动，就能把两者在头脑中建立起联系。[5] 不过，传统做法是把标签放在图标下方。

把标签放到文本字段内部时要多加小心。因为一旦消失（必须的，因为用户得输入内容），用户可能会就忘记这个字段是干什么用的，而且除了单击文本字段外部，没有别的办法再看到标签内容。更好的做法是把标签放在字段上方（“玩具搜索”）或放在按钮上（“搜索玩具”），同时再在字段中放一个示例文本（如“棋类游戏、乐高玩具或玩具娃娃”）。

注 5：要了解更多关注眼睛注视的信息，请参考 J. Edward Russo, “Eye Fixations Can Save the World: A Critical Evaluation”（目光停留能拯救世界：一次重要评估）和“A Comparison Between Eye Fixations and Other Information Processing Methodologies”（目光停留与其他信息处理方法的比较），发表于 *Advances in Consumer Research* Volume 05. 561–570（1978）。

确保每个说明性文案与控件严格匹配。比如，不要文案中说“把商品添加到购物车”，而按钮上写的却是“购买而不添加商品”。

可能的情况下，要给出说明关系的文本，而不要只给出日期和时间。“3 个小时以前”比单纯一个时间戳更容易理解，因为后者会强迫用户心算过了多长时间。（当然，有时候确切的日期和时间是必需的，也不容含糊，见图 3-35。）

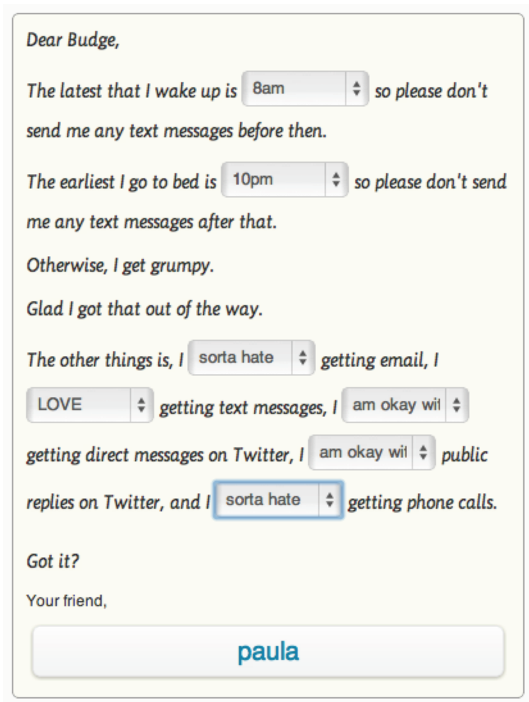


图 3-35：Budge 的待办事项提醒设置屏幕用清晰的文本和选项，把本来枯燥的填表过程变得很有意思（蒙 Paula Te 和“Little Big Details”惠允使用）

除非你有意搅局或误导用户，否则不要使用双重（或多重）否定！这样的话让人摸不着头脑：“如果你不想退订我们的邮件简报，就不要取消选中这个框。”

3.4 算法

1832 年，一个自学成才的鞋匠的儿子宣称："人类大脑能够毫不费力地积累知识……人的大脑依靠某种机制运作。"那一年，他 17 岁。20 年后，这个已然成为大学教授的人出版了一本名著：*An Investigation of the Laws of Thought, On Which Are Founded the Mathematical Theories of Logic and Probability*（《思维规律的研究》）。（与很多名著一样，这本书刚出版时也受到了批评、忽视，人们对它也不屑一顾。）这位教授叫乔治·布尔，是我们熟知的布尔逻辑之父。

布尔发明了逻辑代数，它包括三个基本的逻辑运算：与、或、非。这几个运算构成了生成算法的基础。而算法，用 Christopher Steiner 在他的 *Automate This: How Algorithms Came to Rule Our World*（《自动化：算法如何统治我们的世界》）中的话说，就是：

> 大型决策树由一个接一个的二元决策构成。开车、买卖股票、找对象，以及我们所做的一切，几乎都可以分解成一连串的基于二元输入的二元决策。
>
> ……
>
> 本质上，算法就是表面上能够执行并得到理想结果的一组指令。算法的一端是输入，另一端是输出。

从最根本上说，规则其实就是算法。而有些微交互确实依赖算法运行，比如搜索。搜索框自动填入的内容就是由算法生成的，更不用说搜索结果的排序了（见图 3-36）。商品推荐、驾车路线，以及大多数邮件和读物都是通过算法生成的。某些品牌的元素，比如耐克 FuelBand 的 NikeFuel 点，也是基于算法的，就像 FiftyThree 著名的 iPad 应用 Paper 中的定义颜色一样。[6]

注 6：参见 Chris Dannen 2012 年 11 月 8 日发表在 *Fast Company* 杂志中的文章"The Magical Tech Behind Paper For iPad's Color-Mixing Perfection"（iPad 版 Paper 中"完美混合颜色"背后的技术）。

过去，这些算法都由工程师设计的，但随着越来越多的产品依赖算法，设计师的介入就是不可避免的了。毕竟，再漂亮的搜索微交互如果搜索不到有价值的结果，还是没用的。

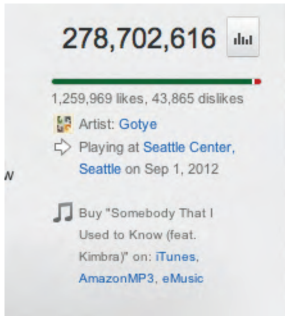


图 3-36：如果你在 YouTube 上观看音乐视频片，它会通过算法匹配你的位置和艺术家的巡演日程（蒙 Nanakoe 和“Little Big Details”惠允使用）

虽然没有办法在这里展示算法背后的代码（因为太复杂了），但展示算法的定义却没有问题。任何算法都由以下四部分构成。

- 顺序

  各个步骤的先后顺序如何？哪个步骤在先，哪个步骤在后？有没有条件，有没有基于特定条件触发的运作？对于耐克 FuelBand 这样的设备，顺序的体现大概是：每两步（根据硬件加速计的计量结果）就加 1 个 NikeFuel。

- 决定

  通常都是“如果……就……”形式的句子。比如，如果时间是 00:00，就重置。

- 重复

  算法怎么循环？循环的可以是整个算法，也可以是部分步骤。例如，当用户在搜索字段中输入时，每输入一个新字符就更新一次搜索结果。

- 变量

  变量是数据的容器，是算法巨大威力的来源。有了变量，就可以不用重写算法而调整计算过程。搜索结果数可以是变量，经过的步骤数也可以是变量。变量可以数值、字符或逻辑值（真 / 假）。

可以通过一个例子来理解算法的这几个部分。比如，有一个显示推荐音乐的微交互，那么顺序就是你想显示什么类型的音乐，以什么顺序显示。这些音乐都属于一个流派吗？新歌显示得更靠前吗？而要做出的决定可能包括：用户以前听过这个艺术家的歌吗？如果听过，就不推荐了。在推荐音乐的列表被填满之前，算法可能需要不断循环。而变量呢，可以是流派、艺术家、专辑、听过、类似、节奏，等等所有可以用来标识音乐的特征。变量还可以是新旧音乐的比例、要显示的推荐音乐总数。

让用户知道算法中操作的是什么数据或变量有时候也很有用，这样他们就可以在可能的情况下手动调整这些数据或变量。比如，知道了耐克 FuelBand 的加分规则，用户就可以适当增加活动量。但现在呢，这个加分办法还是显得有点神秘。当然，有些算法（比如谷歌的搜索算法）是极其复杂的，解释起来不易，特别是通过微文案解释，根本不太可能。

从设计微交互的角度来说，最重要的一点在于知道用户想干什么，哪些数据或内容会最有用，然后把这些对人有益的价值融入到算法设计中。设计算法时只考虑效率而忽视价值的情况太常见了。

最后，规则的问题还是在于它们不可见。用户要理解它们只能等到出现某种恶劣状况时，比如苹果修改了“另存为”，或看到了系统的出错反馈。而反馈正是第 4 章将要介绍的主题。

3.5 小结

规则是微交互模型的非技术层面，规定了什么可以做，什么不可以做，以什么顺序做。

规则必须反映约束。业务、上下文和技术约束都要仔细斟酌。

不要从零开始。以你对用户、平台或环境的了解为基础，然后致力于改进微交互。

去除复杂性，把控件数量减至最少。

减少选项并给出聪明的默认项。选项越多，规则越多。

为每个对象定义状态。这些东西随着时间或根据交互情境怎么变化?

宁肯多花时间提高辨识度。以少搏多。

使用规则预防错误。让用户不可能犯错。

让文案尽量简短。标签够用时，不要使用说明性文本。

协助定义算法。在决定后的代码中体现人类的价值。

第 4 章

反馈



一个 56 岁的家伙举起拳头狠狠地砸向玻璃，玻璃碎了一地，机器的线路全都暴露出来。“是，机器是我砸的，我还会再砸。”他跟保安人员如是说。（这个人后来被判了 90 天监禁。）另外一位，59 岁的 Douglas Batiste 同样因为袭击机器——对着机器撒尿而被拘留。一位妇女因为使劲拍了三下机器，造成 1800 美元损失。[1] 而 67 岁的 Albert

注 1：《纽约时报》2012 年 7 月 13 日“Failing to Hit Jackpot, and Hitting Machine Instead”（没中头奖，砸了机器）（Sarah Maslin Nir）。

Lee Clark 由于其抱怨未得到工作人员的满意处理，跑到自己车里拎出一把枪，然后跑进去对着机器连开数枪。[2]

什么机器让人情绪这么激动？那就是老虎机。

赌博的人每花 10 美元，老虎机就会吸走 7 美元。积少成多，最终达到数百亿，远远超过电影、电子游戏，甚至色情等其他娱乐行业。[3] 作为一个微交互设备，老虎机之所以那么擅长从人们口袋里掏钱，关键在于它提供的反馈。大多数（应该说全部）反馈都暗藏玄机，目的只有一个，就是让人欲罢不能。

什么，你从来没玩过，甚至都没见过老虎机？好吧，老虎机的规则如下：把硬币、美元，或者带条形码的游戏卡（适合新式机器）投进机器，按一个按钮、点一下触摸屏或拉一下控制杆（触发器），然后三个（有时候是更多个）看起来毫无关系的“转轮”开始转动，几秒钟之后，转轮分别停止转动，如果三个转轮对齐后符合某种模式（比如三个转轮上的符号一样），你就中奖了，老虎机就会“吐”出钱来。一个全身投入的玩家一小时可以玩几百次！

事实上，老虎机的规则是不公平的。从统计结果看，它的回报率永远不会超过 90%，因此转轮永远不会“随机”转动；当然，反馈给人的感觉肯定是随机的。假如转轮真是随机转动，那回报率将达到 185%~297%，这显然不是赌场老板想看到的。转轮的输出结果是“加权随机”的。换句话说，空白和低赔付率的符号比大奖符号出现得更频繁。如果真是随机的（而不是看起来如何），那么大奖符号出现的机会要多很多。但是，老虎机给人的反馈让玩家对什么是加权一无所知，而且就算是紧挨着的两台机器面临同样的情况时，它们加权方法甚至都不一样。由于现代的老虎机都是联网的，因此加权方法甚至可

注 2：美联社 2012 年 2 月 13 日“Man charged with shooting slot machine”（一男子冲动之下对老虎机开枪）。

注 3：《纽约时报》2004 年 5 月 9 日“The Tug of the Newfangled Slot Machines”（新式老虎机的明争暗斗）（Gary Rivlin）。

以远程、动态控制。[4]

不管怎么启动转轮（比如，使大劲儿拉控制杆），玩家都影响或改变不了结果。有的老虎机还有一个停止按钮，用于手动停止旋转的转轮。这个按钮也不会影响结果，只是提供了一个控制的假象而已。[5]

除了给转轮分配不同的权重，防止玩家赢之外，老虎机还会刺激玩家产生赌博研究专家 Kevin Harrigan 所谓的“啊，呸”效应，方法就是时不时就停在“差一点就赢了”或者下一次就行了的地方（见图 4-1）。比如，前两个转轮停下后显示的符号一样，但第三个转轮显示空白。这种“差一点就赢了”出现的次数比单纯靠机率出现的次数多 12 倍。研究表明，“差一点就赢了”的暗示会激活人脑中与“赢下来”相关的部分，导致人即使赢不到钱，也会继续赌博下去。[6]

就算有玩家赢了钱，那也是小钱儿。然而，这时候的反馈却格外夸张——灯光四射，声响雷动，让玩家感觉自己好像赢了个大的似的。在《纽约时报》对老虎机设计者 Joe Kaminkow 的介绍中，有一段是这么说的：

> Kaminkow来之前，（老虎机制造商）I.G.T.的博彩程序也配有音效（勉强能说有吧），但却没有充分利用那种有冲击力和爆破感的音乐，即所谓的“智能音效”。Kaminkow决定给每一个操作、每一次转轮转动、每一个结果都配上各自不同的音效。（1999年）Kaminkow刚到I.G.T.时，一般的老虎机大都只有15种“声音事件”，而现在平均每台机器拥有的音效数量接近400种。而且，玩家赌博的时间越长，音乐的速度越快，声音通常也越大。[7]

注 4：《纽约时报》2006 年 4 月 12 日“From the Back Office, a Casino Can Change the Slot Machine in Seconds”（暗箱操作：赌场几秒钟内可以更改老虎机）（Matt Richtel）。

注 5：据 Kevin Harrigan 的“The Design of Slot Machine Games”（老虎机游戏设计，2009）。

注 6：Luke Clark、Andrew J. Lawrence、Frances Astley-Jones、Nicola Gray：“Gambling near-misses enhance motivation to gamble and recruit brain-related circuitry”（失之交臂的暗示刺激赌瘾，激活大脑相关电路）Neuron 杂志，61 卷，第 3 期，481-490 页（2009 年 2 月）。

注 7：《纽约时报》：“The Tug of the Newfangled Slot Machines”（新式老虎机的明争暗斗）（Gary Rivlin）。



图 4-1：你看，“差一点就赢了”（蒙 Marco Verch 惠允使用）

老虎机微交互那么让人上瘾的秘密何在？就在于它持续不断地反馈，持续不断地刺激人的行为。老虎机玩家于是就不断重复相同的行为，不赢钱誓不罢休。就玩老虎机而言，如果花多少钱是可以预测的（比如，再玩一次肯定赢），那玩家很快就会感觉无聊和厌烦。让人欲罢不能的，正是这种投入的不可预测性，加上概率极低的中大奖的诱惑。一般来说，我们不希望在自己的微交互中采用这种强化行为的做法，而只想（通过反馈）提供一种一致的对我们期望行为的正向激励。可预测性是我们期望的。

老虎机给我们启示在于它的反馈极其有效，而且反馈决定了微交互的成败。视觉和听觉刺激组合起来，让原本简单、机械的重复动作——一次又一次地拉控制杆，变成了一种让人充满期待又兴奋不已的行为。显然，赌徒沉迷于老虎机是想赢钱，而你肯定不想让每个微交互都变成闪光喧闹的老虎机。但这并不影响我们的结论：反馈决定了微交互的特质与个性。

4.1 反馈阐释规则

与设计时有意隐瞒规则的老虎机不同，我们微交互中反馈的目的，是帮助用户理解微交互的规则。如果用户按下一个按钮，至少会有两件事发生：按钮被按下了以及按钮被按下后导致什么结果（见图 4-2）。老虎机只会告诉你前一半（控制杆被拉下来了），而不会告诉你后一半（后台到底发生了什么）。因为用户一旦知道后一半，就不想玩了，至少不会再玩那么多次了。不过，由于不用通过反馈告诉用户微交互的实际运作机制（真正的规则），因此其反馈只要能让用户感觉微交互在正常运转就足够了。考虑到对触发器的情境支持，反馈还应该告诉用户能对微交互做什么、不能做什么。

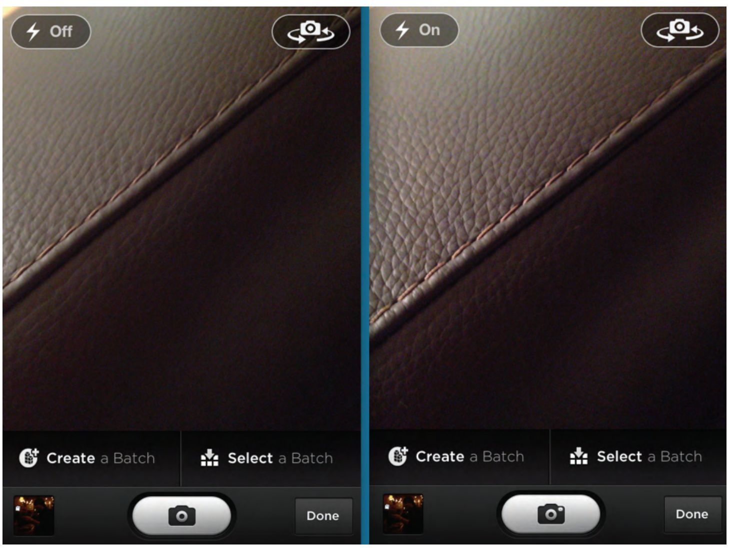


图 4-2：在照片批量上传应用 Batch 中，如果打开闪光灯，快门按钮上的相机图标会显示一个白色的闪光灯指示标志（蒙“Little Big Details”惠允使用）

请大家记住：有时候，不让用户知道规则的细节同样有合理的非欺骗性的原因。每次触发传感器或每次联网获取数据没有必要都让用户知道，而只要向他们反馈一些重要的变化即可。比如，用户没有必要经

常收到没有新邮件的提醒，只要在收到新邮件时提醒一次就好了。微交互反馈的第一条原则是：别让反馈给用户造成压力。怎么判断呢？可以问自己：怎么通过最少的反馈传达同样的信息（见图 4-3）？

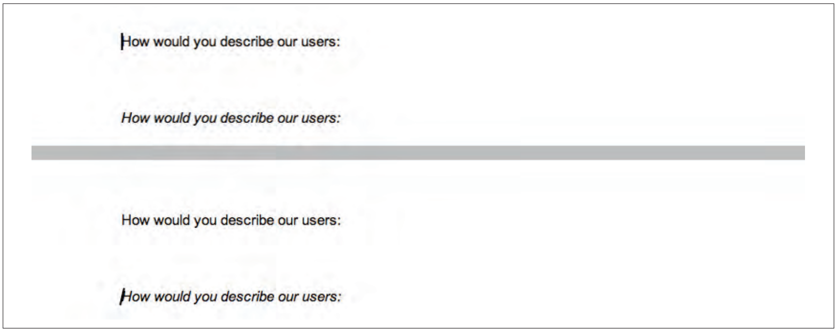


图 4-3：Google Docs 会在用户以斜体输入文字时把光标也变成倾斜的；微软的 Word 也一样（蒙 Gregg Bernstein 和“Little Big Details”惠允使用）

反馈应该由需求驱动：用户需要知道什么，什么时候知道（多长时间一次）？至于反馈的形式，则取决于设计师的想法，视觉、听觉、触觉或它们的组合（见图 4-4 和图 4-5）。

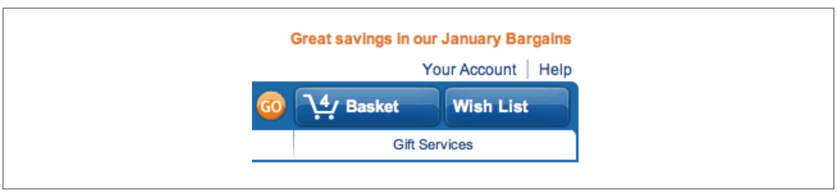


图 4-4：亚马逊在购物车按钮上显示商品数量（蒙 Matthew Solle 和“Little Big Details”惠允使用）

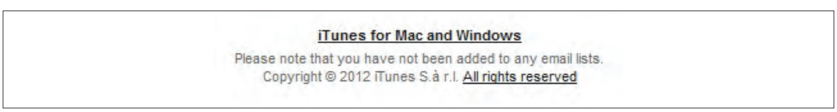


图 4-5：有时候，告诉用户什么事还没有发生也很重要。在通过邮件推荐一款应用时，苹果的 App Store 会告诉你没把你的邮件添加到任何邮件列表（蒙“Little Big Details”惠允使用）

在以下情况下，应该给出反馈。

- 手动启动触发器之后或者手动调整规则之后 / 期间。所有用户发起的运作都应该伴随有一个系统的确认（见图 4-6）。按下按钮时必须表明确实有事发生。

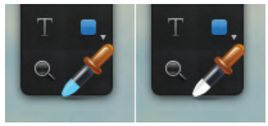


图 4-6：图像处理软件 Pixelmator 的滴管工具会显示你刚刚选取的颜色（蒙“Little Big Details”惠允使用）

- 微交互（或周边功能）的状态因为系统启动触发器而发生显著变化时。显著程度因使用环境而异，必须由设计师因地制宜地决定。有些微交互需要（应该）在后台运行。最简单的例子就是邮件客户端检查新邮件。不一定每次检查新邮件都需要通知用户，但检查到新邮件时一定要通知用户。
- 用户突破（或超越）了规则时。这时候就快要发生错误了。理想情况下，这种状态不应该出现，但有时候也难以避免，比如用户输入了错误的值（比如密码不对），或者已经滚动到了列表底部，再没有什么可以显示了。
- 系统不能执行命令时。比如,微交互因设备未连接网络而不能发短信。这时候需要注意的是，在给出无法完成任务的反馈前，应该先进行多次尝试，比如，尝试多次连接网络。认识到这一点，就能明白为什么要等到多次尝试后才能显示错误消息。
- 在关键过程特别是需要花较长时间的操作中显示进度。如果你的微交互需要上传或下载，最好能估算出完成任务的时间（见图 4-7）。



图 4-7：FTP 文件传输工具 Transmit 4 在一个进度条中同时显示总体进度和个别项目的进度（蒙 Stefvan der Feen 和“Little Big Details”惠允使用）

在以下情况下，可以给出反馈。

- 过程的开始和结束时。比如，下载完了一个文件。
- 模式的开始或结束时，或者切换了模式时（见图 4-8）。

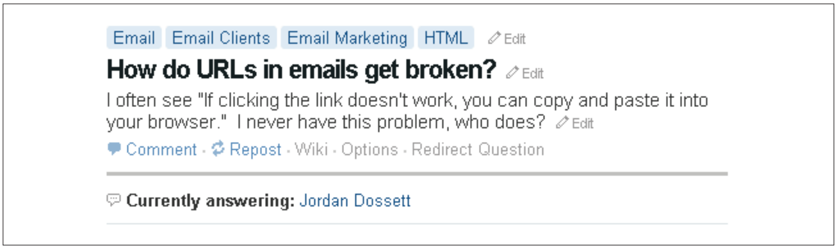


图 4-8：在问答网站 Quora 中，可以看到谁回答了你想知道的问题（蒙 Allison Ko 和“Little Big Details”惠允使用）

尽量通过反馈消除微交互的神秘感。没有反馈，用户永远也理解不了规则。

4.1.1 以人为本

虽然机器之间同样有反馈，但我们最关心的还是与使用设备的人进行沟通。对微交互来说，需要给用户提供的反馈通常有：

- 什么事情已经开始了；
- 用户刚刚做了什么事；
- 哪些过程已经开始了；
- 哪些过程已经结束了；
- 哪些过程正在进行中；
- 用户不能干什么。

知道了要给用户发送什么消息后，唯一的问题就是如何传达，如图 4-9 到图 4-11 所示。反馈的方式完全取决于微交互寄身于什么硬件。如果是手机，有视觉、听觉和触觉三种可能。而如果是其他消费电子设备，可能就只有视觉反馈形式可用，比如 LED 灯。

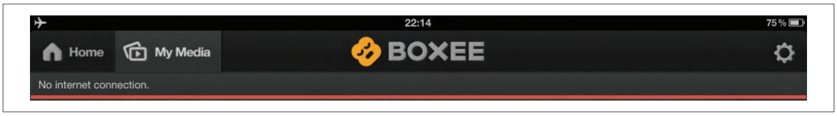


图 4-9：人脸图标是用户喜闻乐见的。Boxee 的标志会在设备未连接互联网的情况下变成橙色“睡觉”的样子（蒙 Emil Tullstedt 和“Little Big Details”惠允使用）

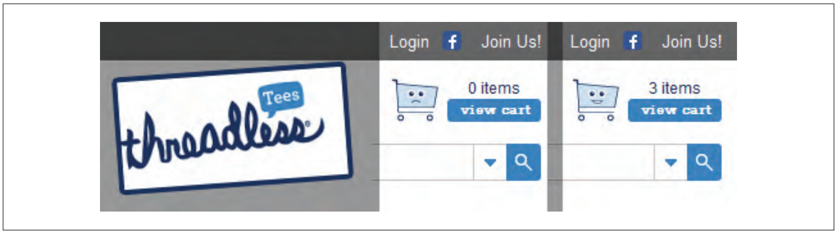


图 4-10：Threadless 购物车上的人脸图标会在你放进商品时由皱眉变成开心地笑起来（蒙 Ahmed Alley 和“Little Big Details”惠允使用）

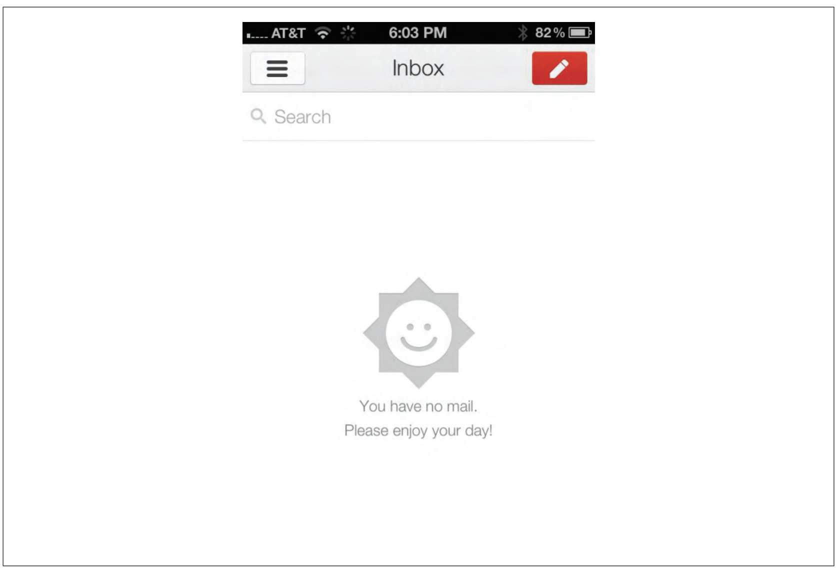


图 4-11：iPhone 上的 Gmail 应用是一个反面教材——连不应该开心的消息都放了笑脸图标（蒙 Steve Portigal 惠允使用）

以洗碗机为例，它的工作过程大致如下：用户选择一项设置，打开开关，开始洗碗，停止。如果洗碗机工作期间机门被打开，它会报警。想象一下，如果洗碗机上有一块屏幕，上述每项操作都可以在屏幕上对应着显示一条消息（比如“正在洗涤，20 分钟后结束”）。如果没有屏幕，那恐怕就只能通过 LED 灯或声音来传达这些消息了。比如，正在工作以 LED 灯闪烁表示，而洗涤完成则打开蜂鸣器。

文字性（即事先写好的）反馈通常并不推荐使用（特别是在没有屏幕或屏幕不够大的情况下）。在舍弃文字形式的情况下（别忘了世界上有相当一部分人不识字，美国中央情报局的资料显示，大约 7.93 亿成年人是文盲），必须诉诸其他传达信息的手段，比如声音、图示、图片、灯光，甚至触觉。既然这些形式都不是文字性的（即使是文字有时候也会含糊不清或模棱两可），那它们就可以向所有人传达消息、传达任何消息。LED 灯闪烁是什么意思？图标变色想说明什么？有些反馈随着时间推移就能让人明白：图标亮了，我一点它，就能看到新短信。点击（或跟随）了任何可能被误解的反馈，都不应该给人“惩罚”。如果我没猜到 LED 灯闪烁表示洗碗机正在工作，打开机门也不应该溅我一身滚烫的热水。事实上，神经学研究发现，错误能提高效率。人类的知识往往是在结果与预期不相符时得到的。

*反馈的第二条原则是：最好的反馈永远不会高深莫测。*它始终向用户传达有用的消息，并且反馈与导致反馈的动作之间一定存在某种深刻的联系。按下按钮启动设备，然后听到嘟嘟声，这没什么意义。因为触发器（按下按钮）或导致的结果（启动设备）与嘟嘟声没什么关系。而如果配上一次咔哒声（按下按钮会发出的声音）或在设备启动的同时辅以某种视觉 / 听觉上的提示（比如某个音调越来越高）就会好得多。难以猜测的反馈很难让人把动作和结果联系起来，因而也就很难让用户理解到底发生了什么。最好的微交互都能做到触发器与规则以及反馈紧密结合，让人感觉它们在一起非常“自然而然”。

4.1.2　少即是多

采用的反馈方式越多，反馈就越具有侵入性。动画加音效加振动，比单独采用任何一种方式都更能吸引人的注意。*微交互反馈的第三条原*

则是：用最少的反馈传达最丰富的信息。先确定要传达什么信息（“下载已经开始了”），然后再选择能传达该信息的最少反馈。信息越重要，反馈方式就要越明显（甚至多路并进），如图 4-12 所示。

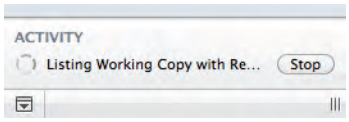


图 4-12：在 Cornerstone 中，旋转轮盘的段数等于后台正在处理的过程数（蒙 usuf Miles 和“Little Big Details”惠允使用）

反馈的第四条原则是：使用被人忽视的东西作为传达信息的手段。很多微交互会包含界面中的某个司空见惯的部分，这样没错。UI 中这些被忽视的部分——滚动条、光标、进度条、提示条 / 悬浮层等，都可以用于反馈信息。如此一来，不用在屏幕上显示新东西，照样能传达出比平时更多的信息（见图 4-13）。比如，鼠标移动到一个被禁用的按钮时，光标可以改变颜色。



图 4-13：OS X Lion 的光标会变化，告诉你不能朝哪个方向调整窗口大小（蒙“Little Big Details”惠允使用）

4.2 反馈体现个性

与讲究实用的触发器和对应规则的控件相比，反馈可以作为一种体现微交互乃至整个产品个性的手段。通过反馈，可以制造一点紧张或幽默的气氛（见图 4-14 和图 4-15）。

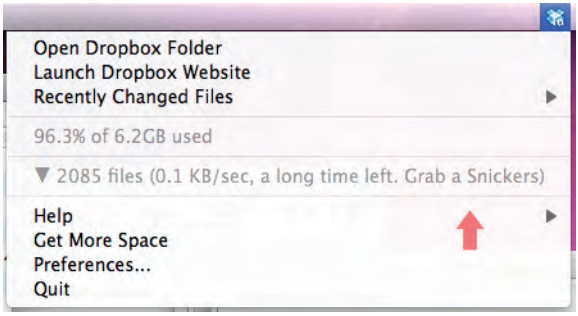


图 4-14：在上传时间比较长的情况下，云存储服务 Dropbox 会建议你吃根巧克力棒（蒙 John Darke 和“Little Big Details”惠允使用）

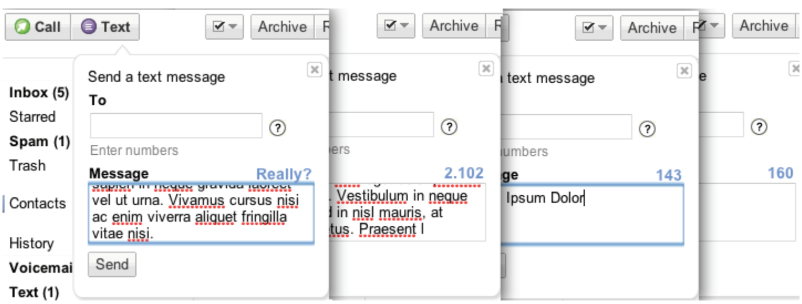


图 4-15：如果你的短信太长，Google Voice 不再计算字数，而是会说“你当真？”（蒙 Zoli Honig 和“Little Big Details”惠允使用）

之所以可以这么做，正如前面所说，主要因为反馈是给人看的。用户对状况做出反应，而我们要对用户给出反馈，哪怕是通过机器。为产品赋予人性，赋予它们所不具备的动机和特质，早已不是新鲜事了。你的笔记本电脑不会故意死机，你的手机也不会跟你生气。利用人类的这种特点，设计师应该有意识地让产品具备某种人格。特别是对于微交互，这种短暂又人性化的时刻不会让人觉得讨厌或无聊，而更可能是讨人喜欢。

比如苹果的语音应用 Siri，你可以说“她”就是个实用工具。没错，“她”能回答的大多数问题都很实用。可是，对于一些意思不明或者“人活着有什么意义？”之类没有真正答案的问题，Siri 的回答是：“我不知道，不过我想肯定有个应用能帮上你。”换句话说，原本可能就是个错误消息（“抱歉，我无法回答这个问题。”）现在却有了几分幽默感和亲切感。的确，出错或者其他可能让用户不耐烦的时刻（比如下载时间很长），都非常适合用个性化的反馈消解用户的情绪（见图 4-16）。



图 4-16：IMDb（Internet Movie Database，互联网电影数据库）网站的 500 错误消息引用的是一句电影台词（蒙 Factor.us 和“Little Big Details”惠允使用）

当然，如果做得不好或者做得过头了，有个性的反馈也会令人讨厌。你应该不希望登录微交互在你每次登录时都提点意见，也不希望在自己忘记密码时会受到应用的嘲讽：“又忘记密码啦？真衰！”你追求的应该只是*表面上*有个性。过于人性化的机器人（所谓的“恐怖谷理论”[8]）可能会让人有面对行尸走肉的感觉；同样，过于人性化的微交互也是有害的。实际上，一点点个性就能百战不殆（见图 4-17）。过度人性化不仅会抬高人的期望值（用户会假定微交互异乎寻常地聪明），而且还可能给人留下不着调、诡异或可恶的坏印象。

注 8：要详细了解恐怖谷理论，请参见 *Popular Mechanics* 杂志 2010 年 1 月 20 日的一篇文章：“The Truth About Robotic’s Uncanny Valley: Human-Like Robots and the Uncanny Valley”（机器人恐怖谷的真相：类人机器人与恐怖谷）。



图 4-17：移动版 Twitter 知道在手机上输入是件麻烦事儿，知道用户在登录时可能会出错（蒙 Joris Bruijnzeels 和“Little Big Details”惠允使用）

说到恐怖，还要提醒大家一句：如果你想随着时间推移，收集并利用行为数据来改进（或创建）微交互（同时毫无保留地告诉用户你在收集什么数据），明目张胆（提供反馈）是最容易让用户讨厌并觉得受到威胁的办法。收集数据实现个性化的目的是让用户高兴，因此最好不要引起用户的反感。

4.3 反馈方法

人类能够通过五大感官来接受反馈，但其中最主要的还是接下来要介绍的三种方式：视觉、听觉和触觉。

4.3.1 视觉方法

必须承认一个现实：大多数反馈是视觉反馈。道理很简单，因为我们都是看着操作，所以给出视觉反馈是最直观的。视觉反馈的形式有很多种，包括表示在哪里可以输入文本的光标闪烁、屏幕文字、发光LED，以及屏幕之间的效果过渡（见图4-18）。

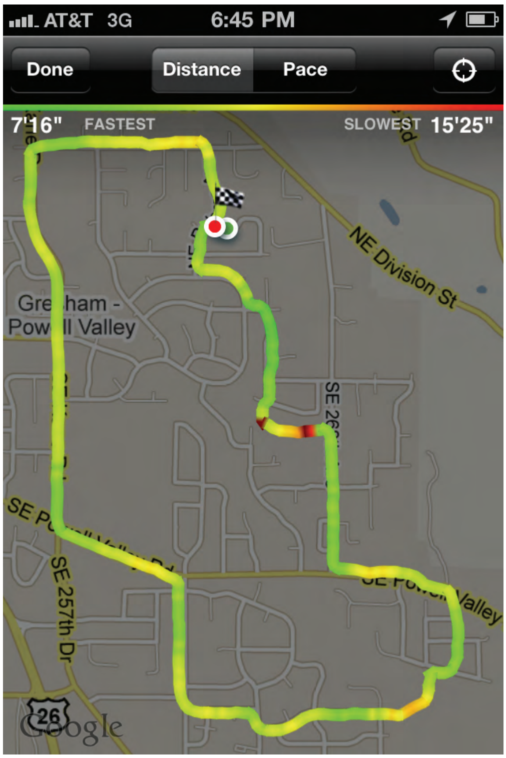


图4-18：Nike+应用在路线图上能显示哪一段跑得最快，哪一段跑得最慢（蒙David Knepprath和“Little Big Details”惠允使用）

除非没有屏幕或LED，否则默认的反馈方式就应该是视觉方式。几乎所有用户发起的动作（不包括用户不能做的动作，比如单击没有预定目标的地方）都应该给出视觉上的反馈。对于系统发起的触发器和

规则，只有一部分应该配有视觉反馈，这一部分指的是那些需要用户介入的（比如，出错提示），以及能够为用户提供操作媒介的（比如，显示收到一封语音邮件的徽章图标）。想一想，要让用户做出决定，需要给他看什么。然后，尽量以巧妙地方式呈现出来。用户通常需要了解资源的情况，包括时间、努力、未读信息，等等（见图 4-19）。



图 4-19：Navigon 应用会在你的车进入隧道时更换背景，而且会显示离出口还有多远（蒙“Little Big Details”惠允使用）

不要显示多余的视觉反馈。比如，千万不要显示内容跟按钮标签一样的提示条。任何反馈都必须力求清晰，而不是增添杂乱。类似地，视觉反馈要适度，即反馈越频繁，越要尽量不干扰用户。除非一定要吸引用户的注意力，否则不要使用闪烁（见图 4-20）。



图 4-20：Github 在提示条里显示时间（蒙 Scott W. Bradley 和“Little Big Details”惠允使用）

理想情况下，视觉反馈应该出现在用户输入的附近或原地。如果“提交”按钮在屏幕底部，就不要把错误消息显示到屏幕顶部。正像第 2 章所说的，人在注意某些东西的时候，视野会变窄。视野外部的任何东西都将被无视（见图 4-21）。如果必须把视觉反馈放到远离视觉焦点的地方，可以通过让它运动（比如，让它慢慢地淡入）来吸引注意力。

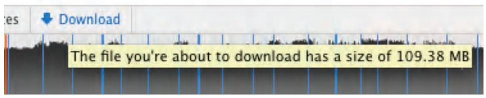


图 4-21：SoundCloud 把下载文件的大小放在了提示条里面，而这种做法只适合大多数用户不介意下载量的情况（蒙 David L. Miller 和“Little Big Details”惠允使用）

1. 动画

人类大脑对运动具有极强的响应能力，因此动画一定不要多用。如果可以不用动画，那就别用。没有动画会让微交互速度更快，认知负担更少。但另一方面，若运用得当，短小精悍的动画既能增添趣味性，又可以传达信息（见图 4-22）。

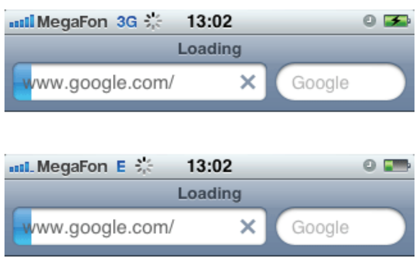


图 4-22：在 iPhone 上，接入方式图标旁边的转轮的转速会随网速快慢而变化。比如，Edge 网络比 3G 网络的转速快（蒙“Little Big Details”惠允使用）

微交互中，动画最重要的一点在于它不仅指示而且精确地指示一种行为模式。如果面板不是从屏幕右侧滑出，就不要让它从左侧滑入；同理，如果用户打开它的方式是向下轻扫，也不要让它从左侧滑入。为

了动画而动画最要命。最好的动画能向用户传达信息：微交互的结构应该注意什么、发生了什么过程，等等。

谷歌的 Android 工程师 Chet Haase 和 Romain Guy 围绕动画归纳了一套 UI 特征（见图 4-23）。动画应该做到如下几点。

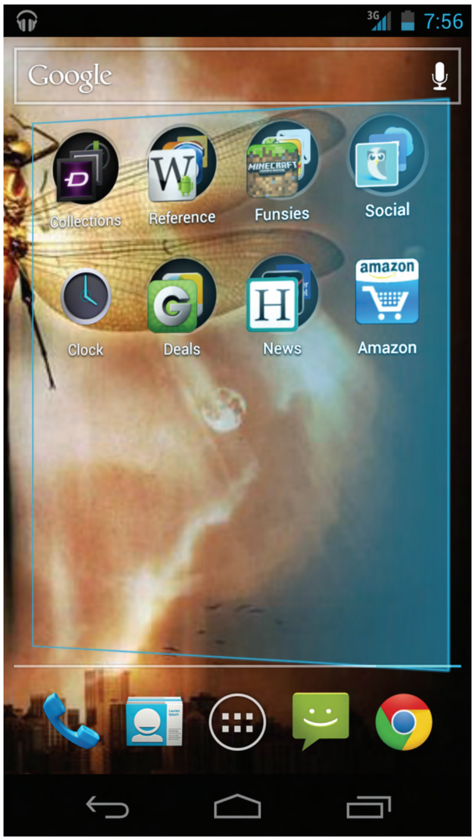


图 4-23：在 Android 4.0（冰淇淋三明治版）中，切换屏幕时，屏幕会产生斜切动画（蒙 Tony Mooch 和“Little Big Details”惠允使用）

- 速度快

  不耽搁时间。

- 平滑
  磕磕绊绊或支离破碎只能破坏气氛，让微交互看起来像有毛病。

- 自然
  看起来要符合自然规则，比如有重力有惯性。

- 简单
  有意义，容易理解。

- 有目的
  不仅仅为了美观。

对最后一点，设计师兼工程师 Bill Scott 归纳了与动画相关的要点[9]。

- 改变视图的时候要保持上下文。滚动列表或平移传送带中的内容，可以看到之前和之后的内容。
- 解释刚刚发生了什么。灰飞烟灭表示有东西被删除了。
- 展示对象间的关系。比如，在拖放结束时，让被拖动的对象移动到放置目标中。
- 集中注意力。修改某项的值时，动画可以让变化更明显。
- 提高感知效能。进度条不会缩短下载时间，但却能让人感觉时间过得没有那么慢。
- 创造虚拟空间的假象。比如，怎么让面板滑入滑出，或者让面板从哪里滑入，从哪里滑出。在用户改变对象状态或切换使用模式时，过渡是微交互动画的重要实现方式。过渡能给人位置感和路线感，让用户知道自己身处何地、路在何方。
- 鼓励深度交互。有意思的动画吸引人交互。

Scott 对动画时间给出有价值的参考值：把任何动画的时间缩短为你开始认为的一半。然后，可能还得再砍掉一半。（详见 O'Reilly 出版的《Web 界面设计》）动画应该让微交互更有效率（通过启发思考或提供指示），甚至让人感觉更有效率。

注 9：Borders.com，2008 年 7 月 15 日："Anti-Pattern: Animation Gone Wild - Borders.com"（反模式：动画也疯狂）http://t.cn/zQLVesU。

2. 消息

设计师 Catriona Cornett 讲述了她升级福特车载 SYNC 系统的经历。在把升级包复制到 U 盘并插到车上之后，她看到了如下升级说明：

> “请严格遵照您车辆上的使用说明。安装开始大约60秒后，您会听到‘安装完成’的提示。此时，不要拔出U盘。必须再等待4~18分钟，等听到第二声‘安装完成’后再拔出U盘。”
>
> 它说会有声音提示升级完成，而实际上并没有。假如我没有看这一小段文字，好像我可能就会做出什么不可挽回的事来。[10]

“安装完成”是一个再明确不过的消息。然而，在上面的说明里，这个短语却让人产生了误解。任何作为操作反馈的消息都必须（至少）保证准确。对于说明性的文字而言，同样应该简洁明了。不要用“错误”“警告”这类没有信息量的词，它们只会搞得用户更紧张。错误消息的反馈文本不仅要说明出了什么错，还要告诉人怎么纠正它。理想情况下，还应该随同消息提供一种纠正错误的途径。比如，不要仅仅告诉用户密码输错了，还要重新把焦点定位到密码框里或者提供一个找回密码的链接（见图 4-24）。

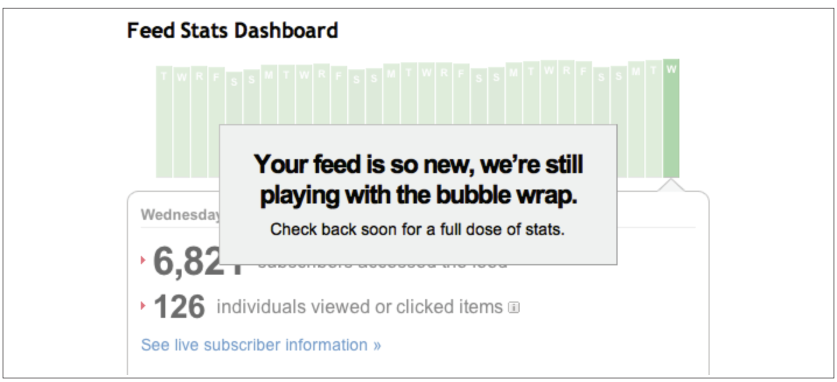


图 4-24：在新闻源聚合站点 Feedburner 中，如果你想马上查看自己的统计信息，会看到这条消息。如果能给出实际的时间就更好了

注 10：2011 年 7 月 11 日：“UX principles in action: Feedback systems and Ford SYNC”（UX 原则实际演练：反馈系统与福特 SYNC，http://t.cn/zQLVesL）。

虽然任何文本都应该直截了当（而且要说人话），但最好不要使用针对个人的语气，比如“你”。“你输入的密码有误”给人感觉更像指责，比“密码错误”还令人不舒服。类似地，也不要使用“我”“我的”，因为它们是反馈文案的恐怖谷。可以像人与人对话一样给出反馈，但微交互可不是人。虽然 Siri 等语音应用使用第一人称能成功，但如果把那些话写下来，你会感觉很不舒服。

理想微交互文案的字数应该以个为单位计算，不能以行，更不能以段为单位（见图 4-25）。文案一定要简短，动词选择一定要谨慎，要把重点放在那些用户可能或需要采取的行动上：“请重新输入密码。”



图 4-25：银行服务仅用短短一行文字（支票将于明天开出，周一快递，7～9 天寄到）就解释清楚了将来会怎么样（蒙“Little Big Details”惠允使用）

4.3.2 听觉方法

第 2 章说过，声音是一种有效的提示方式，能够迅速直达人类大脑，比视觉还要快。由于人类对声音极为敏感（如前所述，对运动的物体也一样），因此要慎重使用。不过，在没有屏幕的设备上，或者作为（用户不会完全注意的）后台运行的微交互的一部分，声音反馈是尤为重要的手段。另外，在看屏幕不安全的情况下，比如开车的时候，声音反馈也很重要。

一般来说，听得见的反馈有两种：强调和报警。用于强调的声音通常意味着用户要主动采取措施，作为对用户期待的任务已经完成的一个确认。单击按钮听到咔哒声就是一个例子。这种情况通常伴有视觉反馈，而且视觉和听觉反馈结合使用已经被证明是比单纯使用视觉反馈更有效的手段。[11] 另一种听得见的反馈，即用于报警的声音，通常意

注 11：参见 Brown、Newsome 和 Glinert 发表于 1989 年的论文“An experiment into the use of auditory cues to reduce visual workload”（通过听觉提示减少视觉负担的试验）：http://t.cn/zQLBmTj。

味着系统发起的一些操作：某个过程结束了、某个条件变化了，或哪里出错了。导航应用中的语音告诉你需要转弯就是声音警告的例子。

微交互中用到的任何声音提示都应该通过雾角测试（Foghorn Test）：这个动作真的重要到用户看不到但应该知道它的地步了吗？即使答案是肯定的，也应该考虑同时提供一个关闭声音的机制。

与其他反馈类似，听觉反馈应该能根据使用环境进行调整。某些 HTC 手机在用户口袋或钱包里时，蜂鸣和铃声音量会变得非常大（手机通过光线传感器能感应到），而在用户把它们拿出来时声音又会变小。有些汽车音响会随着发动机噪音的增大而提高音量。类似地，如果用户不在设备所在的屋子里（通过距离传感器感应）或者房间的噪音非常大（通过麦克风检测），音量和音调也可以增大。另外，如果设备得知用户不方便查看视觉提示（比如开车的时候，通过 GPS 检测），也可以自动开启声音提示（或增大音量）。

音效设计师 Karen Kaushansky 对设计师们提出了建议，建议在设计音效时考虑“没有用的情形”：什么时候声音没有意义？大半夜地在空房间里播放声音——特别是人声，既吓人又扰民。[12]

1. 耳标

声音反馈分两种：耳标和语音。耳标（earcon），是在单词“icon”（eye-con）基础上演绎而来的，指用来传达信息的短促而独特的声音。[13] 然而，耳标能传达的信息有限，因此有时候需要用语音。语音就是录制（口头）的或计算机生成的话音。语音特别适合说明和指示，当然如果产品要面向多个国家，语音的本地化也是个大工程。语音比耳标更费时间，用耳标只要几分之一秒就能传达的信息，用语音就得花好几秒。比较“砰”地一声和“您有一封新邮件！”

注 12：参见 Smashing Magazine“Guidelines for Designing with Audio”（音频设计指南）：http://t.cn/zQLrdbz。

注 13：Meera M. Blattner、Denise A. Sumikawa 和 Robert M. Greenberg“Earcons and Icons: Their Structure and Common Design Principles”（耳标与图标：常见设计原理与结构）*Journal of Human-Computer Interaction*，1989 年第 1 期第 4 卷。

耳标是抽象的，因此要仔细选择与要传达的信息匹配的声音。对微交互而言，最好的耳标能让用户（有意或无意）联想到其他听过的声音。例如，插上门闩的咔哒声可以作为微交互结束的耳标，而向上的嘶嘶声可以作为列表项移动到顶部的配音。耳标不要太尖锐（紧急情况除外），也不要太轻柔（“我听到什么了吗？”）就像在动画里一样，最好的耳标都很简短：不超过 1 秒钟，通常只有几分之一秒。例外是表示正在执行某项任务的伴音，比如用嗡嗡的声音表示正在同步文件。

耳标也必须跟要传达的情绪吻合。这个反馈表示很紧急还是只为了吸引人？这是个警告还是一个通知？耳标的品质（音色、音调、时长、音量）应该与要传达的内容匹配。

如果想让耳标生动鲜明、“过耳不忘”（成为标志性声音），应该至少包含连续的 2~4 个音调（音符）。[14] 你不一定想让自己的微交互被人记住，因此这个技巧即使要用也应该只对每个微交互使用一次。大多数微交互的耳标都应该只包含一个音调，只播放一次。循环播放任何耳标时都必须慎之又慎，哪怕最轻柔、最温和的音调，翻来覆去地播放也会让人怒火中烧。

耳标对一个动作来说应该独一无二。就像不能对不同的动作使用相同的视觉反馈一样，对不同的事件也不能使用相同甚至类似的声响。对于那些与用户动作无关的警告音，就更应该注意这一点。如果用户的目光不在设备上或看不清设备，那他无从知道发生了什么动作。

2. 语音

如果想用语音作为声音反馈，一定要保证口述消息简单明了。如果是让用户根据提示做出选择，给出的选项要明确、简短、少。对于微交互，理想情况下，语音响应应该只有“是”或“不”，最多也就是一个词。正如我们在第 3 章说过的，微交互最好给出聪明的默认值，而不是给出多个选项。任何语音选项都应该出现在消息结尾。“要关闭

注 14：参见之前的脚注和 Joseph Kerman 的 *Listen*（Bedford/St. Martin’s 出版公司，1980）。

声音，说——是”就对了，而“说是关闭声音”则不对。语音反馈之后要紧跟着动作。

使用语音反馈，就要找演员录音，或者使用 TTS（Text To Speech，语音合成）系统。录音能给人亲切感，毕竟是人的声音嘛。不过，就算演员注意通过抑扬顿挫和停顿来准确地传达消息和语调，语音也会包含更多的质感和细微差别。缺点就是只要消息有变化，就得重新录音。

如果消息是动态合成的（比如，建议路线规划指示），那 TTS 可能是唯一的选择。毕竟，没人愿意把不同街道的可能组合都读一遍。虽然 TTS 近年来已经改进了不少，但还是缺少人情味，令有些人讨厌，因此要慎重使用。

4.3.3 触觉方法

触觉，技术上的称呼是“触感反馈”。简言之，触觉就是振动，通常由微电机产生。微电机可以产生强烈的蜂鸣振动，也可以产生模拟平面纹理的更柔和的抖动。与视觉和听觉反馈的数十年历史相比，触觉反馈还算是新生事物，大多数人都是通过寻呼机和手机了解它的。

触觉，因为很大程度上是感觉（在比较硬的表面上振动也能产生噪音），最适合用户贴身携带（手持、触摸、穿戴或背负）的设备。当然，也可以把微电机嵌入家具中，以增强电影和游戏的娱乐性。人的脸和手（特别是手指）的触觉最敏感，而躯干和大腿就差远了。

尽管比视觉和听觉的接触面更广，但人类的触觉（更确切地说是皮肤感觉）也是有限的。不仅受皮肤限制（虽然敏感度不同，但皮肤面积还算比较大），也受大脑限制。负责感知触觉的是四种机械敏感性受体，每种受体检测不同的频率。不同受体之间还会交叉感应，交叉感应的结果就是我们的触觉，因此触觉并不强烈。有研究人员称，我们通过触觉感知的信息只有通过听觉感知信息的 1%。[15] 大多数人都只能

注 15：参见 R. T. Verrillo、A. J. Fraioli 和 R. L. Smith 的“Sensation magnitude of vibrotactile stimuli”（振动触觉刺激的感知量级）*Perception & Psychophysics*，1969 年第 6 卷，P366–372。

感觉到 3~4 个级别的振动。[16] 所以说，通过触觉传达复杂的信息并不可靠。

幸运的是，微交互通常不需要传达复杂的信息。触觉作为微交互的反馈手段，主要有三个用途。第一是强化物理动作，比如在触摸屏上模拟按下按钮的压力，或者在手机关闭铃声时让手机震动。第二是在声音不可用或不适合用的情况下，用来发出警告（这在目前是最常见的）。汽车通过方向盘震动唤醒睡觉的驾驶员就是一个例子。第三是在触摸屏等表面创造人工纹理或摩擦力（这在目前还不多见）。比如，可以让滚动降低速度。

鉴于人类对触觉差异的感知能力很有限，因此它们在微交互中的应用除了细微的、情境化的沟通，就只有干扰性的警告了。除非在音乐家或外科医生使用的特殊设备上，才可能用到差异更多样化的触感。因为在演奏音乐或做手术的时候，更细化的触觉量级可以提供更多的物理反馈。

4.4　反馈的规则

反馈也有自己的生成规则（见图 4-26），这些规则如下。

- 环境变化
  反馈会根据环境的变化而变化吗？比如，到了晚上，音量会加大，还是会减小？
- 持续时间
  反馈要持续多长时间？如何停止？

注 16：Gill, John,“Guidelines for the design of accessible information and communication technology systems”（无障碍信息及沟通系统设计指南）Royal Institute of the Blind，2004 年 及 F. A. Geldard 和 C. E. Sherrick 的“Princeton cutaneous research project, report no. 38”（普林斯顿皮肤研究项目报告，第 38 号）普林斯顿大学，美国新泽西州，普林斯顿，1974 年。

- 强烈程度

  效果有多亮、多快、多剧烈？是用于烘托气氛还是为了引人注意？强烈程度随时间加大，还是保持不变？

- 重复次数

  反馈重复吗？多长时间重复一次？效果一直不停，还是只播放几秒钟？

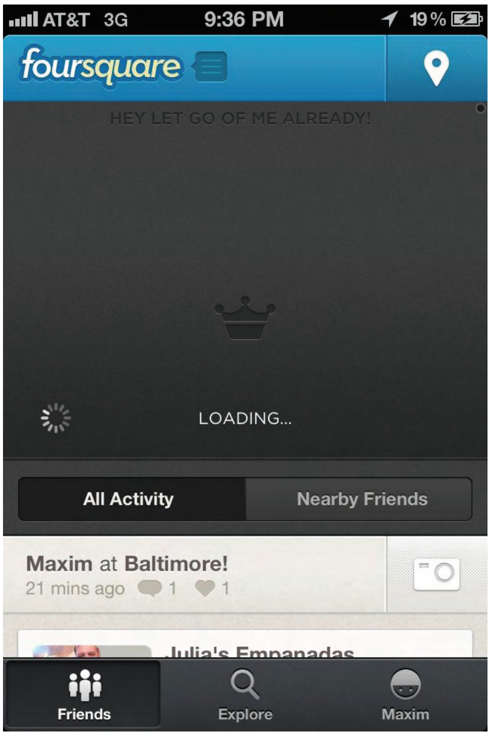


图 4-26：Foursquare 会在你为了刷新而向下拉得过长时报怨一句（嘿，可以放开我了）（蒙 Tory Briggs 惠允使用）

这些规则决定了反馈的各种特点。

如果你不想让用户在玩老虎机时感觉受骗或对着屏幕拳打脚踢，那就

好好设计一下反馈。让用户理解规则，适当时提醒他们状态已经变化。给出的反馈要一致，要鼓励积极的行为。

有时候，一个不断重复的反馈就是一个微交互。在接下来的第 5 章，我们将讨论如何通过循环和模式来扩展微交互。

4.5 小结

掌握用户什么时候需要知道什么信息。所有反馈都以这些信息为基础。

反馈的目的是让用户理解微交互的规则。确定哪个规则值得反馈。

选择需要通过反馈传达的消息，然后再选择正确的一个或多个渠道。

关注情境并关注反馈能否（或是否应该）随之改变。

坚持以人为本。反馈可以利用人性化的外表为微交互增添人性光辉。

使用既有的 UI 元素传达反馈信息。在增加视觉元素前尽可能利用已有元素。

不要随意反馈。把反馈与控件关联起来，或者通过它引导用户行为。

尽可能对用户发起的动作采用视觉反馈。需要强调或报警时，使用听觉和触觉反馈。

第 5 章

循环与模式

2004 年 1 月 4 日，重约 400 磅、有 6 个轮子、使用太阳能驱动的机器人在火星上的一个巨大陨石坑——古谢夫环形山着陆。这个机器人就是造价 4 亿美元的“勇气”号火星探测器，制造时间长达 1 年之久。据 *Passport to Knowledge* 报道，“勇气”号经过 6 个月的太空之旅才到达这颗红色星球，经历了一次危险的着陆，第一次撞击火星表面时反弹高度有 5 层楼那么高。负责建造和控制这个探测器的 JPL（Jet

Propulsion Laboratory，喷气推进实验室）团队认为更糟糕的还在后头。但是，他们错了。

红色的尘埃落定之后，“勇气”号开始拍摄照片并执行科学考察任务，前进到附近的一个目标（“断头谷”）。可是，1 月 21 日，探测任务开始不到 3 周，奇怪的事情发生了。NASA 的深太空网（Deep Space Network）与“勇气”号失去了联系。

一开始，科学家们认为是发生在澳大利亚的雷暴干扰了地面信号接收装置，导致“勇气”号与地面失去联系。但事实是“勇气”号自身出了问题。第二天，“勇气”号发来信号——一次嘟嘟声，表示自己还在那里，然后就再没有其他消息了。问题非常严重。如果是关键的硬件发生故障，那这台机器人就已经死了，它的探测任务已经结束。

在尝试了很多方法后，JPL 最终让这辆火星车发回了诊断数据，时间是 1 月 23 日。虽然很多数据都是没有意义的噪声重复，但也为他们分析故障原因提供了一些依据。结果并不乐观：探测器内部温度超过了预期，电池电量不足。正常情况下，探测器的计算机每天只开 5~6 小时，一方面节省电能，另一方面防止过热。然而数据显示，探测器并没有进入睡眠模式，因此消耗了电能，并导致过热。如果这种状况持续下去，探测器就会报废。这次火星探测活动的负责人 Mark Adler 表示：“我们手里的是一台生病的火星车。‘勇气’号失眠了，还发烧，身体越来越虚弱，满嘴胡话，语无伦次，对我们的问话置若罔闻。”

Adler 扣人心弦的第一手记录可以在这篇文章里看到“Spirit Sol 18 Anomaly”，发表在 Planetary Society Blog，链接为：http://t.cn/zQJXRFn。

黔驴技穷之际，控制人员向“勇气”号发出了 `SHUTDWN_DMN_TIL`（“shutdown dammit until”，关机待命）命令，强制机器人进入睡眠模式。“勇气”号接受了命令，JPL 团队松了一口气。为了确认，他们向火星车发送了一次嘟嘟声：如果“勇气”号真的睡着了，就不会响应。

但他们收到了回音。

“勇气”号并没有执行命令，而且由于地球在环形山中已经“沉没”，除了等到第二天再联系，JPL 没有任何办法。在此期间，“勇气”号会持续过热，耗尽电能。机器人已经生存无望。

JPL 团队重新部署，准备调查清楚到底哪里出了问题。他们提出了一套说法：“勇气”号进入了故障模式，它在遇到自己无法解决的问题时重新启动了，就像我们自己的电脑在死机后重启一样。关键在于，“勇气”号似乎被卡在了故障模式，一遍又一遍地重新启动。换句话说，它在重新启动时碰到了问题。

后来，JPL 发现问题出在探测器飞向火星期间的一次软件升级上。升级过程中要删除旧程序文件，可是代替它们的新文件却上传失败，但这件事没人留意或者说是被忽略了。由于旧文件还在，文件存储空间不足。于是“勇气”号尝试恢复数据，文件系统溢出。为修复问题，“勇气”号重新启动。（这么做也是程序设定的。）但由于文件空间不足，重启无法完成，“勇气”号为此再次重启。这样，机器人就进入了无限循环。[1]

控制团队当时不知道是这个原因，他们只是猜测“勇气”号在不断重启，因此希望在探测器无法挽回地损坏之前停止重启。JPL 怀疑，既然不断重启问题依旧，要么是闪存（即 EEPROM）故障，要么是硬件故障。（假如是硬件故障，那么探测器就不可能修好了。）幸运的是，探测器研发工程师预见到了闪存和 EEPROM 可能出问题，因此设计了一种不通过闪存启动的方式，即通过从地球接收无线电信息，也能执行有限的一些命令。其中一条命令就是告诉计算机不使用闪存重新启动。JPL 团队通过无线电发送了几次这条命令，最终还是起作用了，循环被中止。终于可以长出一口气了，JPL 接收了一些数据，然后让晚睡了很久的“勇气”号进入了梦乡。

还好，故事的结局让人很开心。因为“勇气”号在 2004 年 2 月 5 日恢复了全面运行状态，尽管其太阳能电池板的设计寿命只有 3 个月，

注 1：要了解详细信息，请参考 *EE Times* 2004 年 2 月 20 日的文章“The trouble with rover is revealed”（漫步者的麻烦显现）。

而且还在沙地里被困了两年，但它还是继续运行了好几年。与“勇气”号最后一次通信是 2010 年 3 月 22 日，比 JPL 的预期长了近 7 年。NASA 官方在 2011 年 5 月 25 日宣布“勇气”号完成了使命，因为“在火星上没有多少阳光的严冬，其能量不足以支撑幸存的加热装置，这辆火星车的内部温度比之前在火星上的 6 年都要低”。在本书写作时（2013 年 3 月），“勇气”号的孪生火星车兄弟——晚三周抵达火星的“机遇”号，仍然在执行着探索火星表面的任务。

提到模式，软件应用中也有一个模式与“勇气”号的故障模式类似。在这种模式下，应用以不同寻常的方式运行，通常意味着可以通过比如按一个键来做一些特别的事。而循环呢，与“勇气”号的不断重启类似，就是一个或一串不断重复的命令。（下面会讲到，微交互中的循环会有一点点不同。）正如“勇气”号的惊险故事所揭示的，就算是 NASA 中那些最循规蹈矩的科学家，也很难处理好循环与模式。

5.1 模式

模式是规则的一个分支，而且对微交互来说，模式应该尽可能少用。大多数微交互都应该没有模式，但有时候模式也是必要的。模式最主要的目的是执行一种不常用的动作，这种动作可能会妨碍微交互达成其主要目的。常见的模式就是设置，用户可以在其中指定一些有关微交互的选项。在进入设置模式时，通常仅修改微交互，而不会同时执行主任务。此时的操作与其他操作隔离。比如天气应用（见图 5-1）或股票应用，要在其设置模式中选择城市或股票代码。此时执行的不是微交互的主任务，而是引导用户进入一个子任务，然后再返回主任务的分支规则。

不使用模式的原因往往在于它会导致错误，在模式处于不可见状态的情况下，出错概率更高。比如，打开编辑模式，会把用户原先熟悉的屏幕变成需要重新学习的状态。不同模式下，同样是点击一项，可能会产生完全不同的结果：在默认模式下是选择这一项，而在删除模式下则会删除这一项。模式越少，导致用户困惑的机会越少，在微交互

中，最好没有模式，即使有也不要超过一种，这样强迫用户理解微交互的机会也越少。

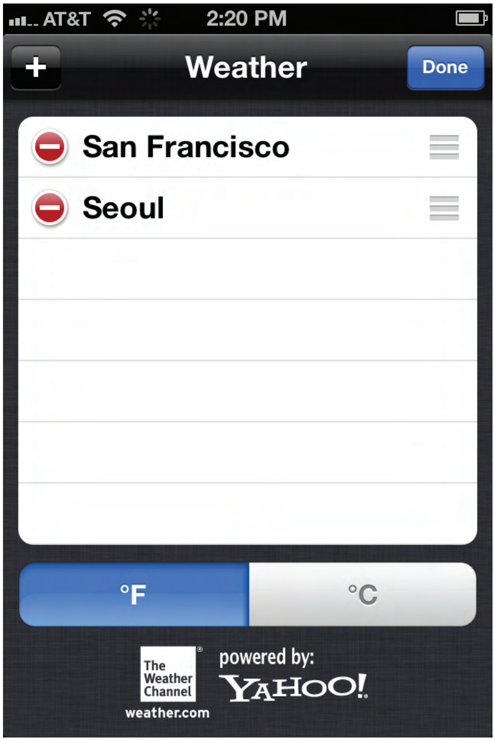


图 5-1：苹果 iOS 版天气应用的设置模式

如果必须设计一种模式，对微交互而言，最好能给它自己设计一个屏幕（当然前提是有屏幕）。（这是第 3 章讨论的“不要让每条规则都对应一个屏幕”的例外。）单独的屏幕可以减少错误和防止困惑，因为用户可以清楚地知道自己进入了一个不同的模式，而不是处于不熟悉的状态。过渡到新模式（以及返回之前的默认模式）的效果，也可以作为有用的提示，表示用户要到一个特殊的地方做一些特别的事。

用户进入另一个模式然后再返回之前的模式时，会希望原来的模式还

处于自己离开时的状态，尽管在另一个模式中操作的结果可能会反映到默认模式。用户在退出特殊模式返回主模式后，会希望主模式中的交互方式与自己离开时的相同，当然要反映出他们在特殊模式下所做的修改。比如，在天气应用中，如果用户在“添加城市”模式中添加了一个城市，那么回到主模式时，应该让用户看到新城市的天气。

弹簧模式与一次性模式

如果切换到另一个模式就是为了执行一个简单的操作，那就会很令人讨厌。除了传统的模式，还有两种适用于（或可以取代传统模式的）微交互模式变体：弹簧模式和一次性模式。这两种模式都不会让用户感觉自己被锁在了一个未知模式里。

弹簧模式（有时候也叫 Quasimode[2]）只在用户按下键或按住鼠标不放的情况下才会激活。用户一停止上述动作，模式就会消失。最经典的例子就是按住键盘上的 Shift 键打开大写模式，但一松开手就会退出。Alt、Option 和 Command 键也都能进入弹簧模式。

弹簧模式的价值在于，用户很少会忘记自己在不同的模式里，因为他们必须通过实际的操作才能切换到另一个模式，而且也不需要切换不同的屏幕。这种模式的缺点是不适合费时的操作或复杂的输入。

对微交互来说，弹簧模式最好少用，它主要适合设备和家电产品。比如，长按“开始”键可以重新启动或重置。

弹簧模式也可以是把用户引导到微交互的不可见触发器，比如自动填充搜索字段。自动填充功能只有在字段里有文本时才会起作用，因此也属于弹簧模式。

这些对一次性模式也适用。*一次性模式*指的是在用户发起模式时，模式持续时间与动作执行时间相等，动作结束，模式就结束。比如，

注 2：Quasimode 是 Jef Raskin 在他后来的奠基之作 *The Humane Interface* (Addison-Wesley Professional) 中提出来的。

在 iOS 中双击文本会调出其剪切并粘贴功能，该功能在选择一个命令后消失。较新版本的 Microsoft Office 中有一个所谓的“迷你工具条”，只要选中文本，它就会出现，其中包含一个格式化工具。在 OmniGraffle 中，每次用完选中的工具（比如直线工具），光标都会恢复到默认状态（指针）。一次性模式最适合快速的任务切换（就像 OmniGraffle）或上下文工具（比如 Office 和 iOS）。

一次性模式也可以用在手势和语音微交互中。在某些语音控制的界面中，比如支持 Kinect 的 Xbox 中，语音命令（在这里就是“Xbox”）就是触发器，一个命令就会启动一个一次性模式，而这个模式又可以接受新的命令。“灯光！暗一些！”或“电视！关闭！”（《星际迷航》里以科幻方式演绎过这种应用：“计算机，定位舰长 Riker！”）类似地，在手势界面中，一个手势（比如挥手）就可以触发微交互，从而进入能够接受另一个手势的一次性模式。在这两种情况下，一次性模式都可以防止意外触发。而在这两方面的例子中，一次性模式都会在一定的时间后退出，为此就需要循环了。

5.2 循环

循环（在微交互的语境下）就是不断重复的一段时间，通常用于设定持续时间。这段时间的单位可以是毫秒、分钟、天，甚至是年。循环的核心是计时，即确定微交互的速度和持续时间。虽然大多数微交互的持续时间很短，但它们或者它们的某一部分可以重复，因而就可能具有不那么短暂的“生命”。

循环由规则（直接或间接地）指明。“每 30 秒获取一次数据”或“运行 3 分钟，然后停止”或“每 10 天发送一次提醒”都表明微交互中有循环。[3]

注 3：程序员看到这个定义可能会有异议，因为他们通常理解的循环（作为计算机科学术语）不可能做到这一点。做到这一点的应该是“等待”或“睡眠”命令。

5.2.1 循环分类

我们关心 4 种循环模式。

- 计数（for）循环
  重复既定次数的循环。比如，在显示错误消息前，检查 10 次是否有网络连接。

- 条件（while）循环
  在满足条件的情况下反复执行的循环。如果条件变化或结束，则循环停止。如果有网络连接，则每分钟检查一次 Twitter 的新消息。

- 集合循环
  与计数循环类似，这种循环会遍历集合中的每个值，然后停止。例子：对每封未读邮件，给未读计数器加 1。

- 无穷循环
  循环开始后，除非出错或被人强制停止，否则永远不会结束。与"勇气"号火星探测器的故事类似，这种循环通常是要尽力避免的。不过，某些类似开灯的微交互基本上就是从一个无穷循环开始的：除非整个微交互被关闭或者灯泡烧掉，否则灯永远不会关。

除了以上循环，还有两种循环：开放循环和封闭循环。微交互将这两种循环用于不同的情况。*开放循环*不响应反馈，它们执行完就结束。("每天晚上 10 点，把灯打开。")*封闭循环*具有内置反馈机制，因而可以自调整。比如，汽车开动起来后，封闭循环可以自动检查发动机噪音，据此调整汽车的音响。

与第 2 章讨论的算法类似，确定循环的参数对用户体验影响极大。循环的次数太少，会让人感觉仓促或有侵入性；而循环时间太长，又会让人感觉拖沓或反应迟钝。图 5-2 展示了一个计时循环的例子。

可以用循环来控制动作时间或让某个过程不会过长，甚至可以控制整个微交互的持续时间。为什么要控制时间呢？可能是出于安全考虑，比如登录银行网站后，如果一段时间内不活动，就会自动注销你的登

录。但这种自动结束微交互的操作有时候会让人反感，所以必须慎用（见图 5-3）。

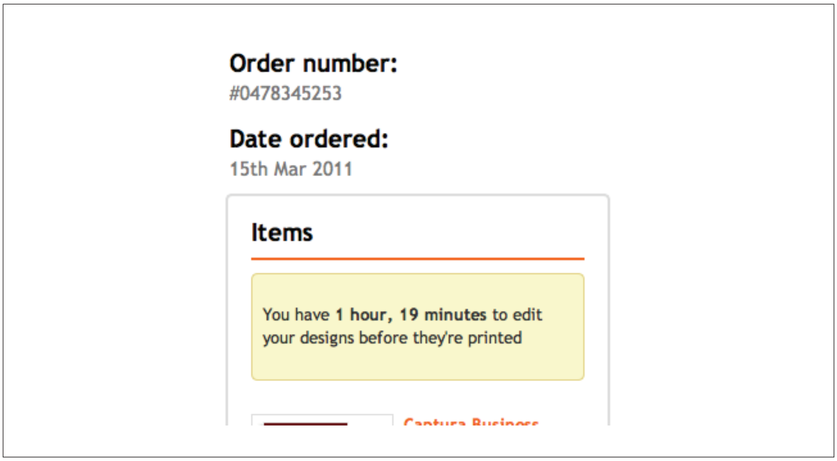


图 5-2：Moo（moo.com）在用户下单后开始计时循环，告诉用户还有多长时间可以编辑刚下的订单（蒙 Matt Donovan 和“Little Big Details”惠允使用）



图 5-3：如果用户短时间内单击了太多次“添加朋友”按钮，Facebook 会对用户进行警告（蒙 Alfie Flores Nollora 和“Little Big Details”惠允使用）

循环也可以用来识别行为。比如，如果用户暂停微交互的时间过长，可以提示用户是否需要帮助（见图 5-4）。

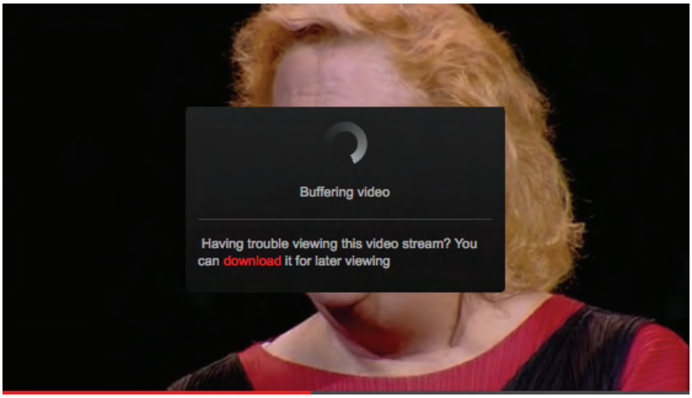


图 5-4：如果缓冲视频的时间过长，TED 站点会给出下载链接，以便线下观看（蒙 Justin Dorfman 和“Little Big Details”惠允使用）

有时候，只有重复（开放循环）还不够。最有用的循环，是那些在长周期内一直存续的循环，以及（或者）能够随时间的推移，根据行为反馈自适应的封闭循环。这就是长循环。

5.2.2 长循环

> “如果对象还没有成熟到令人满意的程度，设计就有问题。”
>
> ——Deyan Sudjic

说说羊角锤吧。跟大多数工具一样，羊角锤在某些特定的情况下非常有用——就像微交互。具体来说，羊角锤可以砸钉子，也可以起钉子，偶尔还可以用来粉碎东西。但微交互不是羊角锤，它有记忆，能使用数据，还可以循环，甚至无穷循环。设计微交互时，可以使用我所说的长循环，不仅关注个别的任务，也关注更长时间跨度内的活动。为了让用户第二次使用微交互时感觉更好，可以做些什么？第十次呢？

第一万次呢？图 5-5 和图 5-6 展示了两个着眼于近期改进的例子。

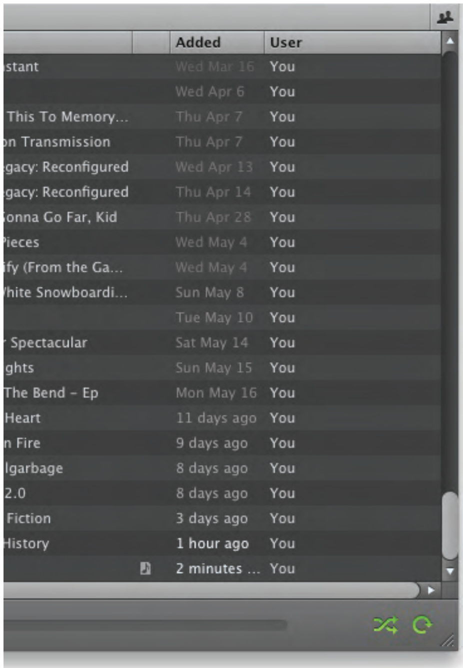


图 5-5：数字音乐服务 Spotify 的“Added”列中的音乐，会随时间的推移而淡出（蒙 Jorge Nohara 和“Little Big Details”惠允使用）

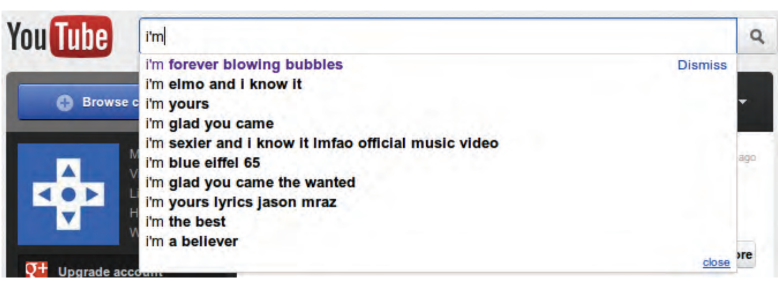


图 5-6：YouTube 会将最近的搜索显示为紫色（蒙 Davide Dettori 和“Little Big Details”惠允使用）

循环可以让你创造出设计专家、Adaptive Path 公司 CEO Brandon Schauer 所谓的“The Long Wow”（长久的赞赏，http://t.cn/zQUOqWe）。“长久的赞赏”意味着随时间的推移向用户交付新体验或新功能，而不是一成不变，借以建立用户忠诚度（见图 5-7 和图 5-8）。从微交互角度讲，“长久的赞赏”意味着随时间的推移调整微交互，让用户感觉它是专门给自己定制的，甚至是一个全新的东西。这就要求使用长循环，也就是持续时间与微交互所在设备一样长，甚至更长时间的循环（如果能远程存储行为数据的话）。

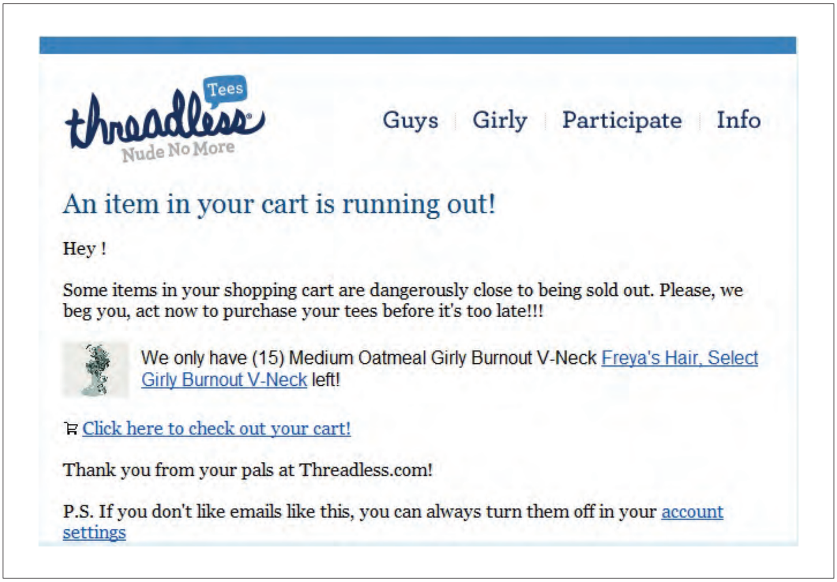


图 5-7：Threadless 会在用户的货物快要卖光之前给用户发邮件（蒙“Little Big Details”惠允使用）

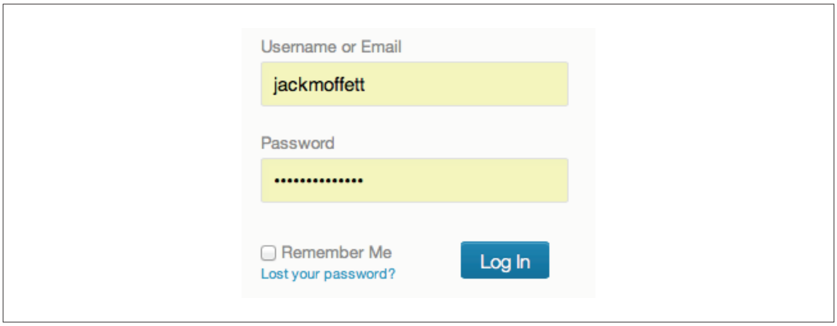


图 5-8：Remember Me（记住我）会打断互联网循环。选这个复选框会发生什么？能保证你自动登录吗？（不可能。）多数情况下，它甚至都不记得你选过这个复选框，所以干脆就不要选（蒙 Jack Moffett 惠允使用）

类似长循环的用途就是扩展微交互，使其不局限于一次使用。比如，每周提醒用户一次愿望列表中的商品，或者看起来没有网站会记住的“记住我”复选框，都是这样的例子。在用户返回微交互时，应该记住用户之前的使用情况。喜欢大声放音乐的用户可能需要与其他人不同的设置（见图 5-9）。

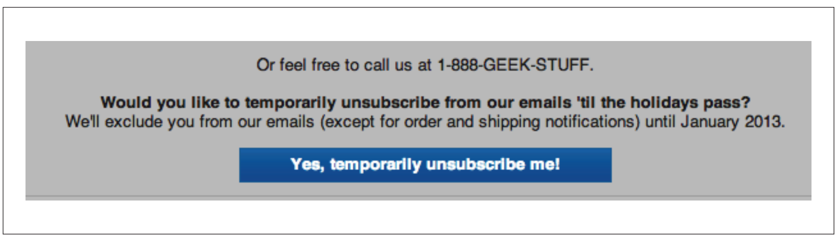


图 5-9：ThinkGeek 允许用户在节假日到来时临时取消订阅（蒙 Kayle Armstrong 和“Little Big Details”惠允使用）

渐进揭示或减少

长循环的另一个用途是在长周期内*渐进揭示*新功能。随着用户对产品越来越熟悉，就不必再手把手地教他们了，可以将他们看成有经验的人。比如，用户多次使用微交互之后，可以加入快捷方式或更高级的功能。

另一个方向是渐进减少（http://t.cn/zY5nBPZ），即微交互随时间推移越来越简单，因为随着用户越来越有经验，一些用于辅助说明的标签就不再必要了（见图 5-10）。不过，必须与否要因人而异。如果因此导致用户使用频率下降，那么微交互必须重新变得可靠和明显。简化微交互的好处，对用户而言就是使界面更清爽、让人更有成就感，而且利用了自己对产品的熟悉程度。

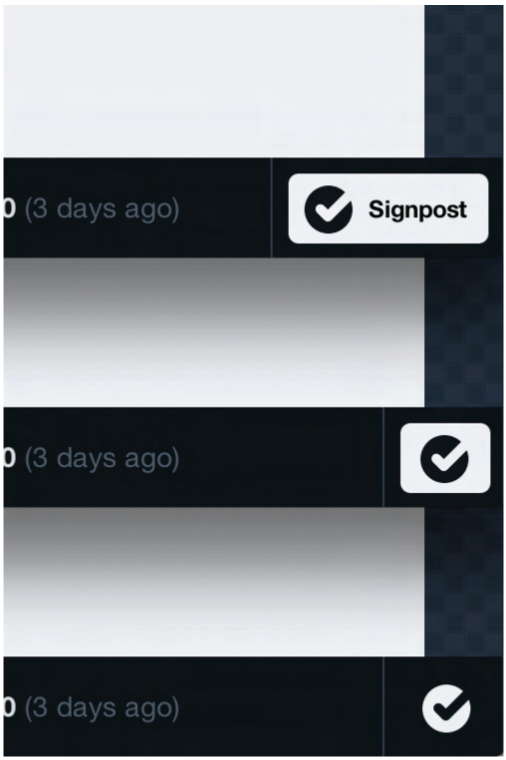


图 5-10：LayerVault 中渐进减少的例子。默认情况下，“Signpost ”按钮有一个大图标，还有一个标签。随着用户越来越熟练，标签就被去掉了。而对更有经验的用户，按钮根本不用再强调（蒙 LayerVault 惠允使用）

差一点损毁的火星探测器“勇气”号的故事告诉我们，循环与模式并不容易实现和维护。然而，利用循环和模式可以让微交互更简洁（通过把不常用的动作，如设置，转移到另一个模式）、更丰富（通过随时间推移不断适应用户习惯）。

现在 ，构成微交互的各个部分就全部讲完了。接下来，该把所有部分组合到一起，看几个实际设计微交互的例子了。

5.3　小结

只有在某个动作会干扰微交互时才需要设计新模式。

如果必须有新模式，尽可能给它设计一个专用屏幕。

对于能够快速完成的动作，可以考虑弹簧模式或一次性模式，而非传统模式。

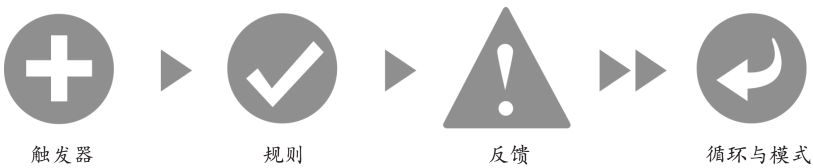
使用循环扩展微交互的生命周期。

仔细确定循环的参数，以保证最佳的用户体验。

使用长循环赋予微交互记忆力，或者使其能随着时间推移渐进揭示或渐进减少新功能。

第 6 章

综合实例



2008 年，波士顿的一个寒冷夜晚，Leah Busque 和丈夫发现爱狗 Kobe 的狗粮没有了。他们打算出去吃饭，路上居然还打到一辆出租车。不过，狗也得吃饭呐。她想：“要是能在网上喊一句：‘谁有狗粮，多少钱，我们给！’而且恰好有个邻居正在超市买东西，于是就帮我们买了狗粮，那该多好啊！”出租车还没到地方，她就把域名想好了：RunMyErrand.com。[1]

RunMyErrand 最终变成了创业公司 TaskRabbit，Busque 成为创始人兼 CEO。TaskRabbit 让人们把自己不想做的事，比如捐旧衣物或买狗粮，外包给周围的人。2011 年，TaskRabbit 获得了数百万投资，员工达到 35 人，每月业务收入达 400 万美元。

注 1：见 *Business Insider*，2011 年 10 月 27 日，Alyson Shontell 的“Founder Q&A: Make A Boatload Of Money Doing Your Neighbor’s Chores On TaskRabbit”（创始人访谈：在 TaskRabbit 帮邻居打打零工就能赚大钱）。

TaskRabbit 的核心就是一个微交互：告诉潜在的“TaskRabbits”有什么任务，然后通过 TaskRabbits 出价。说明需要做什么事就是微交互。而整个服务就建立在这个关键，同时可能又乏味的步骤上。一开始的时候，这个微交互就是一张由许多文本框组成的表单，在这个表单里，用户需要填写清楚他们任务的每一个细节（见图 6-1）。但到了 2011 年，在开发团队设计出更简单的移动应用后，他们意识到还有更好的方法，即设置一组聪明的默认值、提前展示数据并把任务化整为零。正如文章“TaskRabbit Task posting forms”（http://bit.ly/178y67D）中详细介绍的，用户体验总监 Sarah Harrison 解释说：“随着时间推移，我们有了很多任务数据，软件系统也更完善，有条件给任务分类。这样就可以根据任务类型设计特定的表单，通过询问相关细节、设置聪明的默认值和隐藏无关的问题来简化过程。”

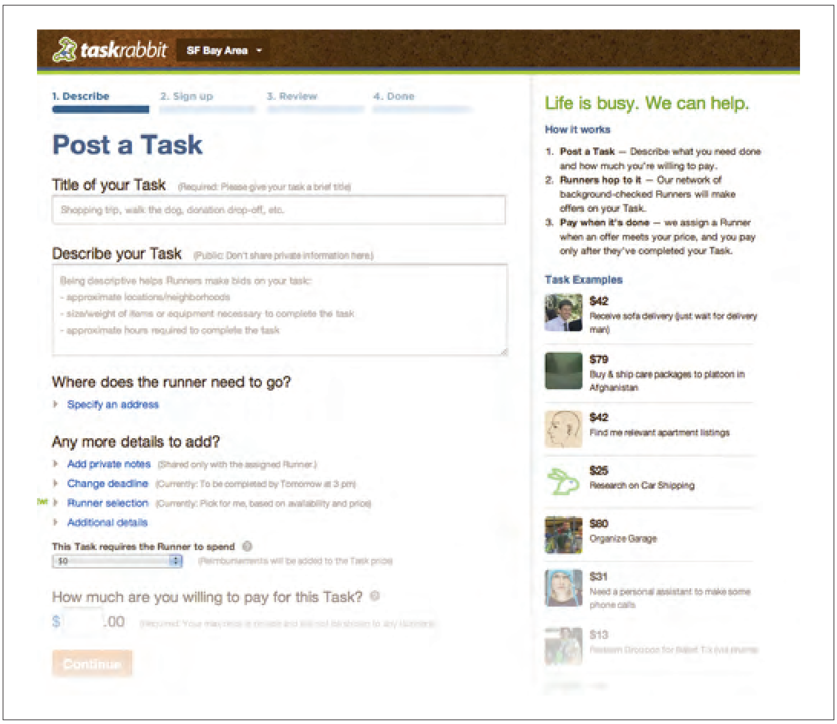


图 6-1：TaskRabbit 早期的任务表单（蒙 Sarah Harrison 惠允使用）

结果就是如图 6-2 和图 6-3 中所示的（无疑，也是个大型的）微交互。用户只需选择一个主任务（见图 6-2），然后微交互的下一步（见图 6-3）就是调整和修改。用户很高兴。“他们把创建任务变得易如反掌，”一位用户如是说，“他们在我想提问之前就给出了答案。”这是一个优秀微交互的标志（见图 6-4）。

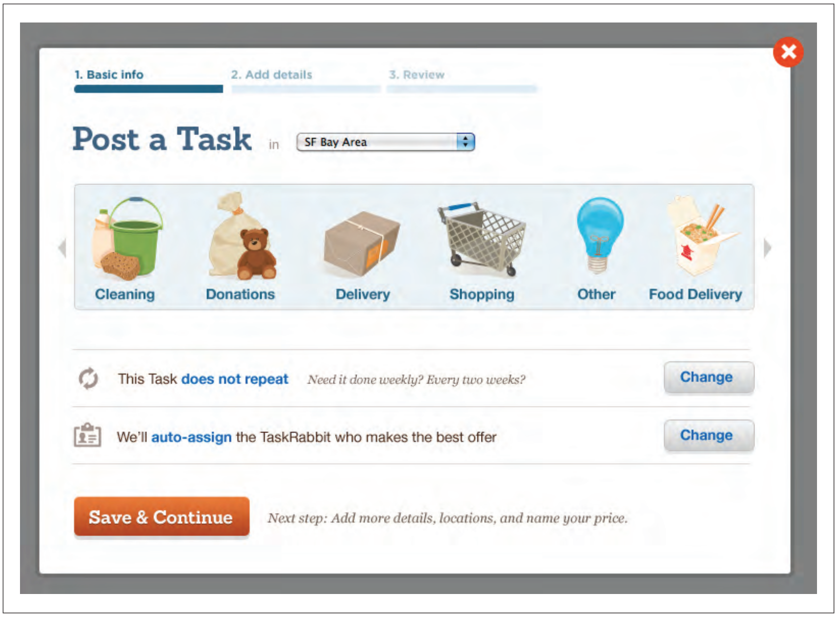


图 6-2：重新设计的 TaskRabbit 微交互的第一步（蒙 Sarah Harrison 惠允使用）

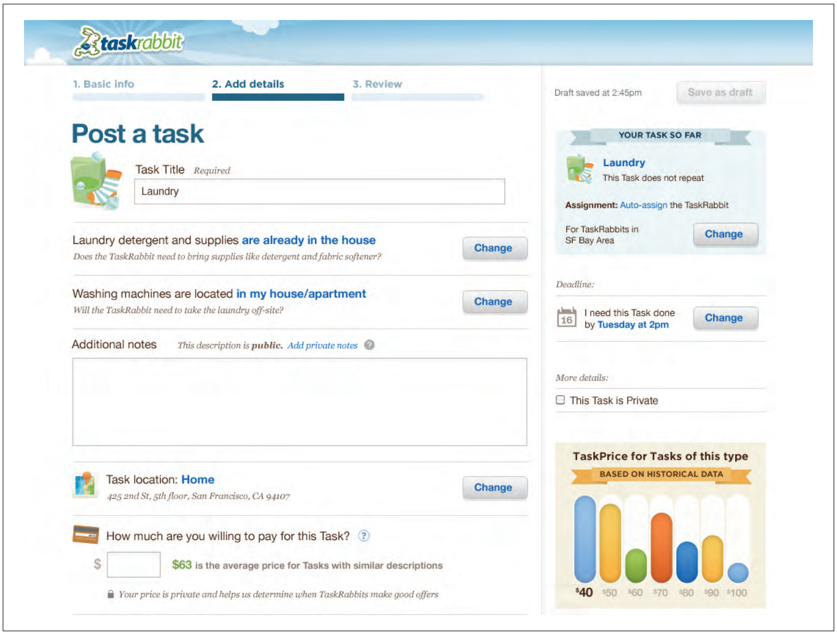


图 6-3：填写表单的第二步。用户选择了主任务后，微交互的其他步骤就围绕这个主任务进行定制（蒙 Sarah Harrison 惠允使用）

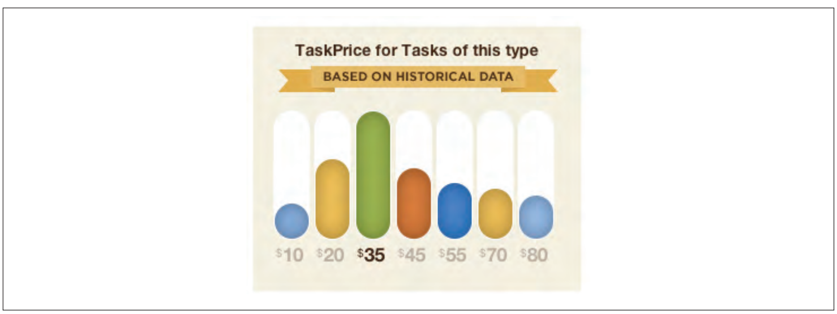


图 6-4：TaskRabbit 提前展示数据，回答“我该付多少钱？”这个问题（蒙 Rishi Shah 惠允使用）

本章，我们要把此前讨论的各个环节综合起来，形成三个微交互的实例：一个用于设置闹铃的移动应用、一个用于分享播放列表的 Web 小工具和一个洗碗机的控制面板。

6.1　例1：闹铃移动应用

第一个例子是一个用于设置闹铃的 iPhone 应用。这里的应用就是一个微交互，应用所要做的就是设置闹铃时间。

为此，首先要考虑这个应用的目标是什么？在指定时间（通常是起床的时候）响铃。这不是设置闹铃，而是一条规则。因此，我们把此时此刻知道的规则写出来：

1. 用户选择响铃时间；
2. 指定时间到，响铃；
3. 用户关闭铃声。

后面我们还会再补充扩展这些规则。接下来再看一看触发器。

既然是 iPhone 应用，那触发器就是主屏幕上的一个图标，没什么可以考虑的了。解决了这个问题，那我们再看看怎么提前展示数据。

用户在启动这个微交互之前，希望看到什么信息呢？在这种情况下，应该是自己是不是设置了闹铃，以及设置的响铃时间。苹果公司为在应用内显示信息提供的机制是角标。于是，我们就遇到麻烦了。在其他 OS 中，比如 Windows Mobile 中，可以使用实时标题，因而可以用文字来写明是否设置了闹铃以及时间。但在本书写作时（2013 年 3 月），iOS 6 只允许在角标中显示数字，而且只允许显示 4 个角标。怎么办？如果闹铃时间是 6:30，可以在角标中显示 630，那要是 12:30 呢？ 1230 能表达清楚吗？而且这样使用角标也不合规则，因为角标里的数字通常都用来表示未读信息的数量。到这里我们有点犹豫不决了。另外，如果真用角标表示时间，那就把自己限制为只能显示一个闹铃时间，不能显示多个闹铃时间了！这不一定是坏事：只有一个闹铃时间可以让规则更简单。一个角标倒也可以显示多个时间，那就是只在角标中显示下一个闹铃时间。但这样也会导致困惑，而困惑是微交互的大敌。如果不确定，就更要简单。因此，在这个限制解除或者开发其他平台的应用之前，唯一能提前展示的就是设置的闹铃数量。虽然看不到时间，但直观地看到设置了几个闹铃还是有点价值的。

我们的触发器还需要一个标签，在这里就是应用的名字。就叫它 AlarmD 吧。

当用户触发 AlarmD 时会发生什么？应用会启动，但启动后显示什么？如果目标是在指定时间响铃，那要么应该显示闹铃什么时候会响，要么应该提示用户设置闹铃。

此时，应该先停下来，问问自己知道用户的哪些信息，以便不从零开始。我们知道用户使用的平台和设备，因此知道可以使用哪些传感器（相机、麦克风、加速计、罗盘）。我们知道时间（当然）和位置。如果应用之前被使用过，那我们还可能知道之前设置的闹铃时间，以及用户使用它的频率。用户每天都在同一个时间设置闹钟，还是每周设置一次？我们还可能知道闹钟响铃后用户会怎么做：用户按没按“再睡一会儿”？如果按了，按过多少次？下面我们根据这些信息再添加几条规则。

1. 如果用户连续三天同一时间都设置了闹铃，则在 AlarmD 启动时提示用户设置相同时间的闹钟。如果用户确实按照提示设置了闹铃，则每次启动都给出同样的提示，除非用户没有选择，那么重置。
2. 如果手机所在国家使用 24 小时制，则使用该格式。
3. 显示用户设置的所有闹铃。在到点之前都显示剩余时间（例如，“还有 8 个小时”）。
4. 用户选择一个时间，则相应闹铃终止。
5. 设置时间到，闹铃停止。
6. 如果用户按了“再睡一会儿”，则 5 分钟后再响铃一次，随后的多睡时间都比前一次少 1 分钟，最短为 1 分钟。
7. 如果使用一周后，若用户没按过一次“再睡一会儿”，则在响铃时去掉该按钮。
8. 用户关门铃声。

注意，这里已经出现了一个长循环（“连续三天”）需要使用系统触发器，以及一些短循环（“再睡一会儿计时”）。显示“还有 8 个小时”是为了防止人为错误，即防止用户过早地设定闹铃，比如让用户只能

选择“下午”而不能选择“上午”。去掉“再睡一会儿”按钮虽然可以减少选项，但仍然可能有争议。因此有必要在设置里提供选项恢复显示的按钮。要是不在 iOS 平台上，那我们可以在用户多睡一会儿后使用其他技巧，比如后续响铃的声音一次比一次大。但 iOS 不允许应用控制全局音量，因此我们也无计可施。

接下来再看一看控件。用户必须能设置闹铃、取消闹铃、关闭闹铃、选择再睡一会儿。这些都需要在用户界面中提供可见的控件，除非其中有些功能可以用多点触摸的手势来实现。除了再睡一会儿，其他几项功能都是基本功能，而再睡一会儿的用户并不清醒，因此不能指望他们做出比按按钮更复杂的动作。设置闹铃时间是这里面最复杂的一个，其他几项都可以通过简单的点按按钮完成。设置时间的方式有很多种，比如使用内置的转轮（像苹果的时钟应用那样），或者自己写一个控件，比如让用户在模拟表盘上选择时间。

此时，我们又该暂停一下，想一想是否把设置闹铃时间作为微交互的标志性时刻，还是只要迅速完成就行。考虑到市场上的闹铃应用大约有 1000 种，所以设置闹铃应该加入一些创意和乐趣。我个人一直非常钟情于老式火车的翻牌计时器，因此打算在这个应用里将其派上用场。老实说，谁会把闹铃设置精确到分钟呢，因此分钟的翻牌步长设置成 5 分钟即可。优化要着眼于大多数人大多数时间在做什么。

真正赋予这个自定义控件生命的是调整它时给出的反馈。这个计时牌必须在时间牌翻过时发出令人神往的哔啪声，而且牌子翻下来时要像受到重力作用一样。另一个重要的反馈就是闹铃本身。可以让用户使用标准的 iPhone 声音，也可以让他们从自己的 iTunes 库中选择音乐，但一个默认的、自定义的声音是一定要有的。其他要准备自定义声音的地方还有取消闹铃或关闭闹铃，这时候应该是一种非常明确的机械咔哒声，譬如关煤气炉的声音。或许闹铃本身也要淡出，而不是生硬地消失。

最后一件要考虑的事件就是循环和模式。很明显，闹铃时间的设置是一个模式，因为只有一个动作，所以可以使用一次性模式：从列表中

点击选择一个闹铃，打开，设置闹铃，关闭。如果你想让用户能设置多睡多长时间，那就再设计一个传统模式：设置。但我个人建议不要设置模式，只要默认值足够合理即可——至少第一次发布时可以考虑。

说到循环，闹铃可谓集几种循环于一身。多睡一会儿是计数循环（闹铃在 5 分钟后再响，然后 4 分钟后再响，3 分钟后再响，等等），然后是条件循环（每过 1 分钟就响一次）。事实上，闹铃本身就是一个长条件循环，因为闹铃时间一到就会响铃，然后一直响到被手动关闭。当然，我们可以（或许也应该）在规则中添加一个计时器，以便在过了一段时间（比如 10 分钟）却还没人关闭它的情况下，自动关闭。

同样，平台限制我们把循环发挥到极致。否则，我们可以给这个应用再添加一个循环，在闹铃报时前 2 分钟通过相机检测房间的灯光。如果房间光线暗，两分钟后逐渐提高手机屏幕的亮度，以便在闹铃响起时，房间能更亮一些。然而，iOS 应用是不能自己启动的。（这条系统规则是有一定原因的，主要是为了防止滥用。）只响闹铃，功能差得就有点远了。

还有几个内置的长循环：用户重复设置闹铃时的提示，以及满足条件时隐藏“多睡一会儿”按钮。我们（有意）没有提到的一件事，就是用户怎么创建重复的闹铃，也就是循环闹铃。如果考虑这个功能，那么应用的复杂性就会增加很多。不过，其中一些复杂性已经转移到微交互自身，也就是那个从一开始就重复检查用户是否设置了相同闹铃的长循环。还可以给这个循环增加一些小细节，比如检查当前是工作日还是周末，并给出相应提示，但为了保证规则简单，到此为止即可。

这就是我们使用本书介绍的原理设计的第一个实例。下面我们再试一个。

6.2　例2：在线共享播放列表

我们要分析的第二个实例是一个在线音乐服务（当然是虚构的）。假设这个音乐服务中有一项功能，即共享播放列表，用户和自己的朋友

可以通过它互相推荐歌曲。另外，我们还假设这个服务已经非常完善，因此可以分享其他服务甚至 iTunes 等桌面应用中的歌曲。最后，假设除了播放列表之外，还有添加朋友和播放音乐等微交互。

首先还是从目标开始。人们之所以想使用这个微交互，原因有两个：发现新音乐和共享音乐。当然，使用这个微交互背后的原因，可能是希望告诉朋友你当前的心情，或者向别人显摆自己的音乐品味。潜在的使用动机同样很重要。

下面先把基本的规则写下来：

1. 如果收到一首新歌，把它添加到播放列表；
2. 用户向播放列表中添加歌曲；
3. 新添加的歌曲放在列表底部。

这里的微交互有两个触发器：添加歌曲（手动触发器）和朋友远程添加歌曲（系统触发器）。先说手动触发器。如果假设用户能在屏幕上看到播放列表中的歌曲，那么用户怎么知道是否可以向里面添加歌曲呢？因为这个服务支持从任何地方添加歌曲，而且系统也足够智能，可以找到所有人都能听的一个版本，所以让用户把歌曲拖到播放列表中看似靠谱。至于怎么让用户知道，或许可以在播放列表顶部预留一个空位，写上“把歌曲拖到这里来”。等用户把歌曲拖进播放列表后，可以再改标签，比如改成“你正在听什么歌？”或者轮流显示几个标签，比如“今天的主打是什么？”或“今天听起来像什么？”

还需要考虑其他手动触发器吗？如果音乐服务有菜单栏，那还可以添加一个菜单项：“添加到共享播放列表”。当然，只有用户在服务中选中歌曲后，这个菜单项才能起作用。假如没有选中歌曲，就打开一个对话框，让用户在里面搜索（音乐）文件。考虑到要把微交互做得轻松好玩，这样显然很笨拙，而且操作也太麻烦了。因此，我们还是只保留“添加到共享播放列表”菜单项，而且让它在选中当前服务中歌曲的情况下能用。至于服务之外的歌曲，只能拖到播放列表中来。另外，再添加一个快捷键，让经常使用的用户只需选择歌曲然后按个键就能添加，而不用去找菜单并进行拖放。

我们实际上还可以允许用户通过输入歌曲名（或者艺术家的名字）来添加歌曲。不过，触发器越不一样，微交互就越复杂。再者说，输入歌名也不是向播放列表中添加歌曲的非常标准的方式。

把歌曲添加到播放列表，特别是添加来自朋友推荐的歌曲时，一定要给出某种反馈，比如利用动画效果。整个播放列表应该下移一个位置，新歌从顶部落下，滑入时伴随着悦耳的叮铃声。既然是一款音乐应用，听觉反馈当然是首选。

还有一个朋友向播放列表中添加歌曲的系统触发器。如果你的浏览器的当前标签页正打开着这个服务，当然应该看（并听）到有歌曲进来。如果当前标签页中不是这个服务，那稍微改一改浏览器标签可能也挺好玩，也就是将它作为一个事件指示器。为此，可以用带笑脸的音符作为“角标”，就叫它 Notesy 吧。

如果用户真的拖动一首歌到播放列表中，那系统可能得花点时间查找匹配的歌曲。虽然可以使用常规的旋转图标，但为什么不来点新意呢？反馈可是塑造个性化的好机会。我们要使用被人忽视的东西，在加载指示器里再次使用 Notesy。比如，可以让 Notesy 像是在“四处寻找”，然后找到了就显示笑脸。

如果找到了不同的版本或不太确定的结果，系统可能必须给出多个选项。如果只找到一个匹配结果，而且（或者）艺术家的名字与歌曲名都完全一样，则应该直接把该歌曲添加到播放列表中。否则，就应该给出可能的匹配结果。展示可能的匹配结果就要用到算法了。因为不想让用户太麻烦，所以我们最多只提供 3 个匹配结果。考虑到艺术家的名字正确的可能性更大，可以将其作为算法中的排序因子，即匹配该艺术家的歌曲优先。如果 3 个结果都不正确，可以提供一种机制，让用户再调出 3 个结果来。如果系统找不到任何匹配项，Notesy 可以消失并显示哭脸。

此时，可以考虑是不是允许用户从共享播放列表中删除歌曲了。我们假设（在最坏的情形下）不给用户提供控件删除自己不喜欢的歌曲，再假设用户不能对播放列表中的歌曲进行排序。或许等到产品测试甚

至发布时，我们才会发现这些都是影响推广和使用服务的致命问题，但至少现在能让规则更简单一点。

下面来看看规则变成什么样了。

1. 如果有新歌到来，把它添加到播放列表中；在浏览器标签中显示 Notesy。
2. 用户可以把歌曲拖动到播放列表顶部或通过选择歌曲再使用“添加到播放列表”菜单项（或者使用键盘快捷键）来添加歌曲。
3. 添加了新歌后，搜索匹配结果。如果找到了匹配的歌曲，以三个一组的形式显示给用户，让用户选择正确的一个。如果没有（找到）匹配的歌曲，显示哭脸 Notesy。
4. 把新歌添加到播放列表的末尾（顶部），并按照添加时间排序。

我们这个微交互还能从哪些方向改进？有什么数据值得提示展示？好吧，如果能知道播放列表中的歌曲是谁的，以及它们加入播放列表的时间肯定更好了。播放列表的总时间和总数量也是有用的信息。另外，能给添加歌曲的人发送有关该歌曲的简短评论（“这可不是 80 后们的歌！”）大概也是一个可以考虑的微交互。

使用一个长循环来鼓励用户维护歌曲是吸引用户的重要手段。为此，可以显示用户贡献的最后日期和时间，作为一个间接的提醒，或者干脆直接通过 Dropbox 标签向用户伸手：“快来添加歌曲吧！”

微交互到什么时候结束呢？实际上，用户登录这个服务的时间就是它生命的上限。当然还应该限制一下播放列表中歌曲的数量，30 首歌好像比较合理。

这个微交互到这里就讲完了。下一个实例，我们讲一讲微交互设备。

6.3 例3：洗碗机控制面板

最后一个实例是设计一个廉价洗碗机控制面板，面临的挑战是洗碗机没有显示屏。假设这是一个非常简单的洗碗机，有一个发声的喇叭和

一个设置不同洗涤时间的装置。再假设我们知道大多数用户想要选什么时间，而且选项不多，只有 4 个。

洗碗的目标是清洁盘子、玻璃制品和银器。因此，基本规则如下：

1. 用户把盘子和洗涤剂放到洗碗机里，然后关上机门；
2. 用户选择洗涤时间并启动洗碗机；
3. 洗碗机洗盘子。

触发器在这里非常重要，稍后我们还会提到。首先，我们看看自己都知道些什么，以便*不从零开始*。我们应该能知道时间（洗碗时长）、用户的上一次设置和用户什么时候做出了什么选择等历史数据。既然是廉价洗碗机，那其他传感器就不可能内置了（设置内部还是有基本的传感器）。因此，可以说它是个哑设备。我们尽管可以不从零开始，但所能知道的也非常有限。

可以提前展示的数据包括洗碗机是否在工作、工作了多长时间和还有多长时间洗涤完毕。大多数人都不太关心洗碗机工作了多长时间，但都想知道还有多长时间洗涤完毕。由于没有显示屏，必须想别的方法来传达这个信息。或许我们可以在这里*使用被人忽视的东西*。

下面考虑控件。我们知道（至少）有两个控件：洗碗机开关和洗涤时间设定装置。开关就用一个按钮即可，而每个洗涤时间也可以分别用一个按钮设定。这样设计操作起来肯定简单：一个按钮全都搞定，或许配合一个点亮的 LED 灯或环绕按钮表示用户选择了哪个时间，让另一个点亮的 LED 灯或环绕开关按钮表示洗碗机在运转。但这些控件无法帮我们*提前展示数据*。为此必须再添加一种显示装置，用于表示洗涤工作什么时候结束，比如一个细条状的 LED 灯，开始时点亮，结束时熄灭。

另一种设计思路是使用旋钮，与洗衣机上的装置类似。用户通过旋转旋钮设置时间，然后拉出或按下旋钮启动洗碗机。随着时间推移，旋钮同步旋转，洗涤结束后停止。更进一步，可以利用旋钮与机体间的缝隙，甚至旋钮上的开槽作为 LED 计时器。与一排按钮相比，一个旋

钮看起来要简单得多。

然而，旋钮常常令人讨厌，虽然我们的洗碗机价格不贵，但我们不希望它让人讨厌。旋钮通常是凸起的，但我们希望洗碗机的控制面板是平的，以免用户不小心碰到。而且，人们使用洗衣机时通常关心它已经工作了多长时间，但使用洗碗机则更多想知道还有多长时间洗涤结束。不要反馈用户不关心的信息。因此，我们就设计一排按钮（也许是那种可爱的电容式按钮），每个按钮对应一个洗涤时间。从左到右，先是最长时间的按钮（炊事用具），一直到最短时间的按钮（快速清洗），然后再隔开一点是“启动”按钮。每个按钮的标签是对应的时间适合清洗的餐具（或“启动”）。在时间按钮下方，可以放条状 LED 灯。

如果要用一句话来描述我们的微交互，既要有意义，还要体现出名词和动词，该怎么说呢？用户选择时间按钮并点亮 LED 光条，然后按下“启动”按钮，LED 光条开始倒计时。看看这句话里的名词，其中每个按钮都有两种可能的状态：被按或没被按。看起来相同的元素其行为也应该相同，因此我们要给每个被按下的按钮加上一圈柔和的光环，当然时间按钮的光环与“启动”按钮的光环颜色可能不同。但是给每个时间按钮都设计一种颜色，这可能就有点过度设计了。根据“提示用户最可能采取的下一个动作”原理，在用户按下某个时间按钮后，“启动”按钮应该想办法吸引用户，因为按“启动”是流程中的下一步。

考虑到 LED 光条显示的是到洗碗机停止工作为止的倒计时（属于计数循环），其颜色应该与“启动”按钮的一致。而且，这个 LED 进度条也可以分成几段，每段大约代表 15 分钟——我们可能无法知道确切的时间，因为有些清洗选项（比如“自动清洗”）的时间取决于内部传感器。如果洗涤完的水还很脏，那可能就会再洗一遍。

到现在，我们还没有考虑洗碗机停止工作时怎么给出怎样的反馈，这个反馈其实非常重要，而且也是微交互的规则之一。别忘了这个微交互的目标就是清洗餐具，清洗结束、目标达成时，用户应该知道。洗碗机有个喇叭，因此一种方案就是在清洗结束时播放“标志性声音”

（比如“嗒哒”之类的）。但不能假设用户一定能听到，而我们也绝不想循环播放这个声音，直到用户打开机门或重置时才停止。（哎，什么“重置”啊？别急，一会儿我们会说到。）好，就让工作期间的“启动”按钮和 LED 光条显示为红色，工作结束时 LED 光条熄灭，“启动”按钮的光环变成绿色（考虑到色盲用户，或许变成蓝色更好），表示清洗完成。“启动”按钮到现在已经有了 4 种状态：关闭、按我、工作和完成。只要用户打开洗碗机的机门，就重置回关闭状态。

现在说一说重置。要知道，有时候洗碗机可能不会像我们期望的那样顺利工作。洗碗机正在工作的时候，用户可能会打开机门，而且一直开着。我们或许可以运用*防呆原理*，在洗碗机工作期间把机门锁住，但这样做好像限制得有点过了。为此，我们需要一些有关开门的规则和一个重置洗碗机的方法。可以单独设计一个重置按钮，因为这个按钮与其他按钮作用不一样，而我们不想让按钮看起来相同但行为不同，所以这个按钮看起来也应该跟其他按钮不一样。另一种方案是在“启动”按钮上添加*弹簧模式*，即按住“启动”按钮不放触发重置。我更喜欢这个方案，因为如果没有更好的理由，少一个不常用的按钮总比多一个更好。这样就可以用更好的名词实现更多的动词。不过，这样也会让重置操作不够明显。为此可以在下面添加一个标签：“按住重置”。另外，可能还要添加一个循环，在机门打开超过一定时间后自动重置。

现在唯一的问题就是我们怎么做到*不从零开始*。可以记录用户上一次选择的洗涤时间以及什么时候做的选择，但这些信息有没有用还不太清楚。没错，我们可以让洗碗机保持上一次用户选择的时间，按 4 个时间按钮计，这样可能会节省用户按时间按钮时 25% 的时间（前提是使用 4 个按钮的机会均等，但这是不可能的）。好像可以在这里添加一个长循环，判断哪次按下的按钮可以不保存。但这样又会导致微交互的行为不一致：有时候某个按钮的灯会自动点亮，有时候又没有灯亮。要么记住最后一次按了哪个按钮，要么就都别记。

以下是我们考虑所有细节之后设计的规则。

1. 用户把盘子和洗涤剂放到洗碗机里，然后关上机门。
2. 除非“重置”，否则使用上一次的时间计时，LED 进度条点亮，“启动”按钮闪烁（“按我”状态）。
3. 用户可以重新选择洗涤时间，进而修改 LED 进度条的计时时长。
4. 用户按下“启动”按钮，“启动”按钮显示红色光环（“工作”状态）。
5. 洗碗机洗盘子。LED 进度条倒计时。
6. 如果机门被打开，暂停计时。在机门重新关上时，恢复计时。如果机门打开超过 1 小时，重置。
7. 洗涤完成后，时间按钮和进度条的灯光熄灭。“启动”按钮显示绿色光环（“完成”状态）。
8. 机门被打开时，“启动”按钮切换为“关闭”状态。
9. 任何时候，如果用户按下“启动”按钮并保持 3 秒钟，微交互重置，洗涤作业停止。所有按钮恢复到关闭状态，进度条计时清零。

这里并没有使用内置的喇叭给出额外的反馈。（假如它是个高端家电，或许还可以使用触觉反馈。）但实际上，的确有几个时间点需要通过声音来强调动作。特别是，如果我们使用的是电容式按钮，可以用声音来模拟按钮被按下时的音响效果。按“启动”肯定要使用耳标，而且必须是“标志性声音”。同样地，洗涤结束时似乎也应该配一个耳标，但这个声音如果半夜在一个空房间里响起，会不会让人讨厌呢？假如我们设计的是一台造价更高的洗碗机，那就可以考虑用程序检测时间（通过网络）、房间亮度（通过光线传感器），甚至房间里的物体移动（通过运动传感器）。这样，就可以只在用户没有睡觉并且就在附近的情况下播放耳标了。可惜，我们设计的不是这一款。

好啦，我们几个微交互的例子到现在都讲完了。希望这几个实例分析能让读者理解本书介绍的有关微交互的结构和法则。

6.4　微交互的原型与文档

任何产品的文档和原型都是为了达到一个目的：告诉别人它怎么（或

应该怎么）工作。对微交互而言，最难传达的是整体流程：所有细节是怎么联系到一起的。微交互的整体流程决定了它给人的印象。

要向人传达整体流程，主要有以下几种手段。

- 构建原型
  如果你的技术能力足够或者有这个条件，那么在目标平台上构建产品的原型，应该是最容易让人理解微交互流程的手段。不过，这种方式可能也最花时间。

- 拍成电影
  电影可以传达时间概念和工作流程。可以使用 AfterEffect 等视频和后期制作软件来编辑（见图 6-5），也可以做成动画，比如使用 HTML5。



图 6-5：原型影片的静止画面。左侧的暂停按钮可以把一个暂停图标“发射”到屏幕上（蒙 BERG London 惠允使用）

- 多帧故事板
  还可以通过一组相互联系的故事板来展示微交互（见图 6-6）。虽然这种方式不能准确地展示时间长短，但至少可以演示某个使用情境下不同状态之间的变化。

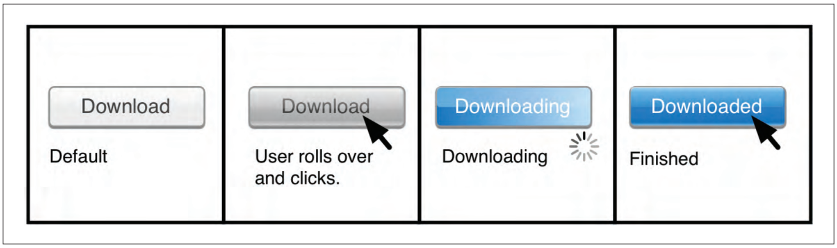


图 6-6：多帧故事板的示例

解释微交互最差劲的方案可能就是屏幕截图了。屏幕截图只能传达很少的微交互流程信息，而且常常不能帮助人们理解它的上下文信息。最好的文档要讲故事，通过情节告诉人们发生了什么和为什么会发生。

要了解更多关于以故事为中心的设计方法，请参见 Braden Kowitz 的文章“Why good storytelling helps you design great products”（为什么讲好故事能帮你设计出优秀的产品）：http://t.cn/zTV6KF3。

如果必须得使用屏幕截图或线框图，别忘了一定要包含关键帧。关键帧是动画片中的一个概念，指的是由资深动画师绘制的关键画面（所以才叫关键帧）。而这些关键帧之间的过渡画面，则留给那些初级动画师去填绘。对于微交互而言，关键帧可能包括触发器、规则中的某个重要时刻和微交互的结束画面。

一般来说，多种方法并用可以更好地解释微交互（http://t.cn/zQHdcRa）。比如用原型或电影表明时间概念，用多帧故事板展示细节和使用情境，而用带关键帧的线框图来说明复杂的规则。

6.5　编排微交互

除非是很特别的应用或设备，否则微交互很少有独立存在的。更多情况下，微交互会处在一个较大功能的周边、内部，甚至核心。比如本章开头介绍的 TaskRabbit 的“Post a Task”（发表任务）微交互。

在设计非独立存在的微交互时，第一件事就是搞清楚它与功能的关系。微交互是用于启动功能（登录）、实现控制（视频播放器上的按钮），还是出现在功能里面（格式化工具），或者结束功能（关闭开关）？每种情况下的触发器可能都不一样。第二件事就是确定微交互的存续时间。暂停按钮可能只要应用开着就会一直显示，而格式化工具或许只会在用户执行特定操作时才有必要出现。

然后，就是考虑微交互是否应该成为“标志性时刻”；换句话说，它是否应该让人记住。大多数情况下，答案都是不必，而且应该不必。当然，它应该有亲和力（这是本书所要强调的），但高效率和不费事才是它的目标。特别是在微交互本身就承载了整个产品目标的情况下（比如用户在能使用应用之前首先遇到的登录微交互），更是如此。

6.5.1 微交互与功能

微交互也可以触发*其他微交互*，因而存在一种“花环”效应，即前一个微交互触发后一个微交互，如此环环相扣。比如，打开设备或启动应用（微交互）后，系统触发器可以检查用户上一次使用应用的时间。如果有一段时间了，就可以启动另一个微交互（“欢迎回来，以下是从上次使用 ×× 应用以来的更新！”）。

只要对微交互精心组织，让一个微交互退出后，另一个微交互紧跟着切入，就可以把多个微交互组织成功能。细节全在设计中。

微交互与功能，在这里就像作曲和配乐，要想编排出好的功能，关键在于知道哪些微交互应该突出表现，而不是对所有微交互都一视同仁。有些要重要体现，有些就该作为陪衬。那么反馈也要通盘考虑，以便重点突出，不喧宾夺主。

采用这种设计方式，最好列一个清单，把所有微交互都罗列其中。前提是最好有一个任务列表，或者功能需求说明书，然后据此提炼出清单。有了这个清单，就可以画出一幅微交互地图（见图 6-7），从而明确相互之间的关系。

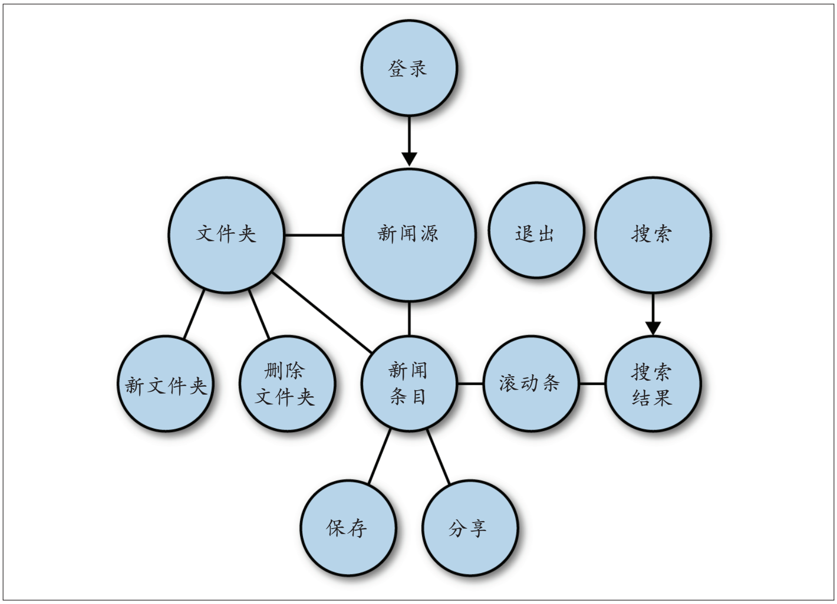


图 6-7：构成新闻阅读器的所有微交互

这里要特别注意“交接棒”环节：哪个微交互触发哪个微交互，在哪里退出一个微交互又启动另一个微交互。事实上，这种交接环节对用户而言可能并不明显，而且很多情况下也不应该让用户察觉。谁也不希望自己的产品有明显拼凑的痕迹，最好的印象应该是一个密切关联的整体。

不过，在微交互的基础上编排功能，是很容易忽略整体的。在设计完每个微交互之后，必须再反过来想：怎样才能让每一个微交互与其他微交互都能融洽相处。特别是在绘制草图或线框图时，很容易在无意间造成两个微交互间的冲突。比如说，考虑某个列表如何显示时，可能会与滚动操作冲突。预防冲突的方法，就是在设计会与其他微交互接触的微交互之前做好标记。

说了那么多，内容基本上都是围绕怎么防止微交互过度设计、过度侵扰用户的。但有时候，还应该考虑怎么防止它们过于呆板，还要给它们注入一些灵性。

6.5.2 改进呆板的微交互

并不是所有人都有机会从一张白纸开始设计。很多时候，我们需要改进产品中已经存在的微交互。当然，还有可能是我们早就开始了主功能的开发，而现在需要考虑怎么让其中的微交互更上一层楼了。从哪儿着手呢？

可以根据本书介绍的原理，向自己提出一系列问题。

- 这个微交互应该作为"标志性时刻"吗？换句话说，应该让用户对它过目不忘吗？要求印象越深，控制和反馈上就越要多下功夫（包括设计个性化控件）。
- 我是从零开始吗？我手头上有没有一些能够用于改进微交互的用户或环境资料？
- 这个微交互中最重要的数据是什么，能不能提前展示它？用户不经意地扫一眼就应该能看到什么？
- 设计一个定义的控件是否合适？自定义的 UI 是昭示当前微交互很重要的关键。
- 我预防人为错误了吗？用户在某些情况是不是可能导致错误，怎么才能自动防止这些错误发生？
- 我使用别人忽视的东西了吗？用户界面或者硬件是不是还有有待挖掘的地方？
- 可不可以给高级用户设计看不见的触发器？是否有哪个环节适合采用隐藏的快捷方式（比如手势或快捷键），从而让用户更快地完成规则？
- 文字和图标都有亲和力吗？微方案听起来像不像一个讲解员的解说词？能不能增添一点幽默？
- 能不能增加一些动画，让界面不那么静态？屏幕或状态切换有没有过渡，有没有指示下一步是什么的（非干扰性的）指示装置？
- 能否开辟出额外的反馈通道？反馈是听觉的还是触觉的？
- 想一想，当用户第二次使用这个微交互时，应该看到什么场景——第 100 次的时候呢？思考一下该使用什么样的长循环。

通过向自己提问，并将答案应用于现有的微交互改进，肯定能让它焕发生机。如果真能如此，本书的使命就达成了。

6.6 往小处想

我们讲座了很多微交互：搅黄马勒《第九交响曲》的闹铃、支持数百万人购买纽约地铁 MetroCard 卡的触摸屏触发器、苹果乱改规则而搞砸的“另存为”、老虎机让人欲罢不能的反馈、几乎断送火星探测器前途的循环与模式，以及宠物狗没饭吃引出的数百万美元的大买卖。小事同样重要，一直会，永远是，而且今天尤甚。

21 世纪的问题多种多样。有些大型、系统的问题不能轻易解决，而有些微小、具体的问题，如果解决得好，则能带给人片刻的安宁、开心和成功的喜悦。我们需要能解决这两类问题的人，特别是能够同时兼顾这两个方面的人，以确保我们设计出来的大型系统——城市、政府、公司、产品，都是以人为本来建造的。而让这些大型系统闪耀人性光辉的，通常都是那些看似微不足道的瞬间，即微交互。在如今这个算法和无人驾驶汽车的时代，人性光辉愈发显得珍贵，应该成为我们上下求索的终极目标。

细节彰显人文关怀、思维方式，以及关注重点。而全人类共同关心的问题，就是我们自身以及自身的需求。通过微交互来满足人类的这个需求，是每一位设计师送给人类的礼物。

往小处想，改变世界。

附录

测试微交互

很多人都觉得微交互没有什么好测试的，甚至说测试微交互无异于问“什么颜色的自行车棚好？”之类的问题。换句话说：这不重要。[1] 但如果你把这本书看完了，那一定会觉得微交互有价值，而且可以通过用户验证、测试和试用来加以改进。

微交互可以利用精益 UX“构建 > 度量 > 学习”的方法体系：构建微交互并测试、通过各种定量和定性的方法度量设计、从对度量结果的分析中学习。然后，再次迭代[2]。

真正的精益 UX 迭代每次都是测试一个“最低限度可靠的产品”，检验概念（假定）是否有价值。而对于微交互来说，我们通常假设整个概念有价值，或者至少对于实现应用或设备功能是必要的。因此，更多的是测试流程和结构，而非概念。另一个与精益 UX 不同的地方体现在原型忠诚度方面。微交互没有最低限度的可测试原型（通常是纸面原型）这一说，因为微交互的结构非常重要，如果想有效地测试它，

注 1：参见“What color should the bike shed be?”（什么颜色的自行车棚好？）：http://t.cn/zQqJLov。

注 2：参见 Jeff Gothel 的《精益设计：设计团队如何改善用户体验》（人民邮电出版社，http://www.ituring.com.cn/book/1031）。

必须把原型做得尽可能与实际产品一样。而且，微交互的触发器、规则、反馈、循环之间联系非常紧密，不容易分割。

在桌面软件中，大多数微交互可能都不会单独测试。单独在桌面上测试微交互涉及的建立和运行测试会话的代价（暂不提为测试而制作原型的投入）非常大。此时的微交互测试通常会与其他测试任务集中到一起进行。在 Web 应用中就不一定了，因为创建 Web 原型很快、A/B 测试也容易，而且分析手段也相对较容易获得。移动应用的原型同样很容易创建。如果整个应用就是微交互，那测试非常重要；构建于一个微交互之上的设备同样如此，只不过制作硬件原型要麻烦很多。

假如你的测试与统计数据有关，那要知道由于微交互很小（因而对它的改动多数情况下同样也很小），必须有很多人参与测试，才能得到有用的信息。多少人才够？至少要几百人（甚至上千人）参与才行，通常 5~8 个参与者肯定是不够的，最少也得 20 人以上才能得到一点稍微有用的数据。如果是定量性数据，那么几百、几千，甚至几万用户都很正常，很多网站的测试就是这样。如果只有 5% 的用户打开下拉菜单，而做出选择的只有 4.75%，那么就算有几千位用户都很难发现这一点，然而这个比例却会极大影响到销售和采用。

除非微交互很难用或很好用，否则想通过定性测试来探知细微差别的有效性几乎是不可能的。定量分析是唯一实际的选择。比如，把谷歌 Analytics 把“活动”添加到一个 Web 微交互中，能够让设计者精准地发现微交互的缺点，而定性方法则必须跟踪很多用户很长时间才可以达到同样的效果。如果你的测试不依赖统计数据，那么即使测试几个参与者，也可以得出结论。

与所有产品测试一样，还应该提防由测试本身导致的所谓的“情境错误”。由于测试是虚构的、人为搭建的情境，测试人员根据安排引导用户的过程中，可能会导致和暴露一些正常情况下不会发生的错误和问题。举一个小例子：中止询问或回答问题，可能导致错过非常重要的反馈。

A.1 测试中的注意事项

测试必须验证以下 4 个最重要的方面。

- 你真正理解了微交互的目标，而不仅仅是过程中的一个环节。设置状态消息的关键不于在字体，而在于沟通。理解这一点可以帮你解决任何强调相关的问题，无论问题在微交互自身，还是在整个产品——这个微交互对整个产品的用户体验有多重要？
- 你真正理解什么数据是重要的。你需要知道什么数据可以提前展示，哪种上下文相关的行为对微交互有用，可以重复使用。
- 某些微方案是否必要，如果是，那它是否清晰易懂。这既包括说明性的方案，也包括标签。
- 时间与流程。微交互的运行时间是不是太长了？循环是不是太长了？或者，是不是太短了？时间跨度太长的长循环很难测试，除非你想做长线调查，而大多数开发人员是不会这样做的。

前两个方面的信息可以通过对话和访谈来收集。第三个方面可以通过观察，而通过观察可以得出结论的事情还有很多，下面仅举一些例子。

- 单击 / 点击 / 按控件的操作是不是太多了？换句话说，用户操作起来是不是太费事了？这不一定是针对点击数，尽管次数也是衡量费不费事的一个指标。
- 用户会不会感到莫名其妙？如果用户说过（真的说了或者皱眉、表情困惑）“为什么要干这个？”那肯定哪里不对了。常见的问题是标签名写错了、没加说明文案，或者有方案但没说清楚。
- 怎么回事？说明反馈可能写得不清楚，或者某个标签同样也不清楚。
- 好像没反应啊？要么没给出反馈，要么反馈太不起眼。
- 那个东西在哪儿呢？用户期待与当前界面存在隔阂。可能是标签问题，也可能缺少一个关键的微交互。
- 这是到哪儿啦？过渡或模式有问题。
- 你刚才怎么处理的我的数据 / 内容 / 输入？这又是用户期待与输出不相符的情况。要么调整标签或文案，如果不是这个问题，那可能就是更深层次的微交互问题，即没有实现用户期待的处理方式，或

者说当前的处理方式让用户感觉不舒服。

- *我单击/按下这里，会怎么样？*这说明标签不明确，或者说明方案写得不清楚。
- *我没注意那还有个按钮。*这是视觉分层出了问题。引导微交互的视觉路径不清晰。
- *原来这样也行啊！*某个操作隐藏太深了。常见于多点触摸的手势或键盘快捷键等不可见的触发器。
- *现在该干什么？*与前面的问题类似，引导路径不清晰，特别是没有提示下一步该做什么。
- *我看到了什么呀？*反馈不清楚的结果，通常发生在流程上。添加标签或者把标签改清楚一些，比如使用提示条。还可能说明你显示的数据并不重要。

以上只是定性数据的例子，但定量分析同样可以参考。

A.2 定量测试

有一句格言说（源自开尔文男爵）："不能测量的，也不能改进。"这句话对微交互或多或少也适用。有一个基准（起点），再有一个可以与之比较变化的东西，是极其有用的。以下列举了几个可以测试的数据点。

- *完成率*
  可以完成微交互的用户有百分之几？
- *微交互的总持续时间*
  完成微交互需要多长时间？（一般来说，不熟练的用户完成微交互的时间是熟练用户的5~10倍。因此，要降低这种极值效应，应该使用几何均值，而不是中值。[3]）
- *特定步骤的持续时间*
- *步骤数*

注3：参见Jeff Sauro的"8 Core Concepts for Quantifying The User Experience"（量化用户体验的8个核心概念）*Measuring Usability* 2012年12月11日。

- 单击/点击/选择的数量

  并不是总有价值，但可以让你知道哪一块不够有效。

- 系统错误的数量

  有没有什么地方在用户操作无误的情况下出错了？（在使用实际数据 / 连接测试微交互时经常会发生这种情况。）

- 用户错误的数量

  可以分两类：犯错和误解。犯错是用户理解动作的目标，但执行了不适当的操作，比如输入了错误的电子邮件地址。误解则是用户不理解规则，尝试做一些规则不允许的事情，例如单击没有响应能力的标题。[4]

还可以尝试量化定性数据，比如对：

- 满意度；
- 难度；
- 信心；
- 有用性。

等定性指标划分等级（比如，1~7，1 最低，7 最高）。然而，如果采样数据规模过小，这种量化很难说明问题。

不过，这样至少能促使你改进微交互，并通过再次测试得知是否有进步。或者，你会开发微交互的替代版，然后将它们进行对比（A/B 测试）。同样，要注意采样规则。用户过少会导致某些错误或个人偏好看起来过分显著或不显著。

即使存在统计显著性，实际上也不一定是显著的。使用数据改进设计时，最重要的一点是要知道：数据不是为你设计好的。数据要靠人来解读，并且要代入情境解读。数据本身很少能告诉你问题的原因在哪里。

数据一般是有意义的，但有些时候又应该忽略数据。为什么要忽略数

注 4：要了解关于犯错和误解的更多内容，请参考原载于《美国计算机学会通讯》1983 年第 26 期 Donald Norman 的“Design Rules Based on Analyses of Human Error”（基本用户错误分析设计规则），以及 James Reason 的 *Human Error*（1990）。

据呢？举个最简单的例子：大多数在线广告不会被点击。如果能有0.5% 的点击率（clickthrough rate），那就相当不错了。[5] 既然使用率那么低，那是不是就该删除在线广告呢？ 99.9% 的用户这么想（另外0.1% 的用户在广告公司上班）。可是，删除广告也就意味着关闭网站，因为没有资金来源保障运营。你希望谷歌消失吗？数据不能反映全部事实，因此不能完全相信它。公司和组织的环境，以及用户基数可不仅仅是电子表格中的数字而已。可以把数据作为决策参考，但绝不能将它作为唯一依据。

A.3　微交互测试流程

下面介绍一个微交互的测试流程供读者参考。这个流程肯定不是唯一的方案，但可以作为你的起点。

1. 在向参与者展示原型前，询问他们期待的微交互的工作方式。询问他们过去用没用过类似的功能。询问他们想通过微交互做一件什么事。确认他们在使用微交互之前想知道什么，即是不是有什么信息可以让他们完全不必使用微交互。
2. 让参与者在完全没人帮助的情况下使用微交互。所有定量数据都要在此时，以及之后收集。
3. 随着参与者一步步地使用微交互，让他们说出自己的印象和判断。聆听参与者是否能说出微交互的工作过程（规则）。注意与实际不相符的地方。
4. 询问他们第二天还会不会再用，以及他们希望微交互记住关于他们的什么信息。
5. 最后，再询问他们哪里应该修改。

按照这个流程，应该可以发现并诊断出微交互存在的问题，同时也能验证用户的总体目标和需求。建议大家至少按照这个流程做两次，每次一组不同的参与者。然后，根据这两组参与者的反馈和问题分析改进微交互。

注 5：参见《商业周刊》2007 年 11 月 11 日“So Many Ads, So Few Clicks”（广告那么多，点击那么少）。

关于封面

本书封面有两种麻雀，即英国麻雀（家雀）和美国树麻雀（树雀）。虽然外表看起来类似，但它们却不是同一个属的。

英国麻雀或家雀与欧亚麻雀更接近。它们头圆、体胖、腿短，均长 16 厘米。虽然原产于欧、亚和北非，但广泛分布于世界各地，这很大程度上是由于 1851 年被引入北美造成的。

作为毫无灭绝危险的一个物种，英国麻雀在很多地方被视为害虫。只要是人类定居点，几乎都能看到这种麻雀。用于吸引更罕见野鸟的喂食器通常都能把它们成群结队地吸引过来。英国麻雀雌雄交配，但未必会等到性成熟。而且，有些麻雀会充当繁衍后代的替代者，如果其他配对中的一方死去，便取而代之。英国麻雀把巢筑在人类住宅周围，为了捍卫自己的巢穴，有时候会驱赶其他鸟类。

美国树麻雀与英国麻雀类似，但与美国麻雀（如褐斑翅雀或布氏雀）关系更密切。美国树麻雀体型与英国麻雀相差无几，但头部有暗红色毛冠和眼线。这一点与欧亚麻雀不同，欧亚麻雀的喉咙部位及眼线是黑色的。美国树麻雀肥胖的身体可以帮它们抵御严寒。它们夏季多生活在加拿大北部和阿拉斯加，到冬季时，会迁徙到加拿大南部和美国北部。

虽然名字叫树麻雀，但美国树麻雀一般并不栖息在树上。因为冻原地带很少有树，而在迁徙之路上，也多是灌木和草原。美国树麻雀会通过扑打翅膀摇掉植物松散的种子，也会啄食地上的甲虫和浆果。作为群居鸟类，美国树麻雀的群体规模不大，但比起喜欢吵吵的英国麻雀，它们相对要安静得多。

微交互 细节设计成就卓越产品

平庸的产品与伟大的产品差就差在细节上。作者将通过本书展示怎么设计微交互，即位于功能之内或周边的那些交互细节。你的手机怎么静音？你怎么知道有新邮件了？怎么修改应用的设置？诸如此类的交互细节，既可以毁掉一个产品，也可以成就一个产品。

高效而有趣的微交互，涉及触发器、规则、循环和模式，还有反馈。透过书中生动、真实的设备及应用示例，读者将理解微交互对于塑造产品个性、赋予产品卖点的重要意义。

- 重新思考设计：自下而上，而不是自上而下。
- 了解微交互设计的历史，以及你的产品中哪些算是微交互。
- 深入剖析触发器：用户或者系统怎么启动微交互？
- 学习如何运用图片、声音、振动来帮助用户理解微交互的规则。
- 理解模式和循环的种类，以及微交互反馈的各种手段。
- 通过闹铃、共享播放列表和洗碗机等实例分析，彻底掌握微交互设计法则。

“《微交互》是我期待已久的一本书。我一直都希望看到一本能够在原子级别上，深刻而又简洁地把交互设计讲清楚的书。这本书就是！”

——Jared Spool
User Interface Engineering（uie.com）

Dan Saffer是Smart Design的交互设计总监，他还著有《交互设计指南》、*Designing Gestural Interfaces: Touchscreens and Interactive Devices*和*Designing Devices*。自1995年起，他从事过家电、设备、应用、机器人、网站和服务设计，数百万用户每天都在使用他的设计。

封面设计：Karen Montgomery　张健
图灵社区：www.ituring.com.cn
新浪微博：@图灵教育 @图灵社区
反馈/投稿/推荐信箱：contact@turingbook.com
热线：(010)51095186转604

分类建议　计算机/交互设计

人民邮电出版社网址：www.ptpress.com.cn
O'Reilly Media, Inc.授权人民邮电出版社出版
此简体中文版仅限于中国大陆（不包含中国香港、澳门特别行政区和中国台湾地区）销售发行
This Authorized Edition for sale only in the territory of People's Republic of China (excluding Hong Kong, Macao and Taiwan)

O'REILLY®
oreilly.com.cn

ISBN 978-7-115-33161-8
9 787115 331618

ISBN 978-7-115-33161-8
定价：35.00元

欢迎加入

图灵社区

最前沿的IT类电子书发售平台

电子出版的时代已经来临。在许多出版界同行还在犹豫彷徨的时候，图灵社区已经采取实际行动拥抱这个出版业巨变。作为国内第一家发售电子图书的IT类出版商，图灵社区目前为读者提供两种DRM-free的阅读体验：在线阅读和PDF。

相比纸质书，电子书具有许多明显的优势。它不仅发布快，更新容易，而且尽可能采用了彩色图片（即使有的书纸质版是黑白印刷的）。读者还可以方便地进行搜索、剪贴、复制和打印。

图灵社区进一步把传统出版流程与电子书出版业务紧密结合，目前已实现作译者网上交稿、编辑网上审稿、按章发布的电子出版模式。这种新的出版模式，我们称之为“敏捷出版”，它可以让读者以较快的速度了解到国外最新技术图书的内容，弥补以往翻译版技术书“出版即过时”的缺憾。同时，敏捷出版使得作、译、编、读的交流更为方便，可以提前消灭书稿中的错误，最大程度地保证图书出版的质量。

最方便的开放出版平台

图灵社区向读者开放在线写作功能，协助你实现自出版和开源出版的梦想。利用“合集”功能，你就能联合二三好友共同创作一部技术参考书，以免费或收费的形式提供给读者。（收费形式须经过图灵社区立项评审。）这极大地降低了出版的门槛。只要你有写作的意愿，图灵社区就能帮助你实现这个梦想。成熟的书稿，有机会入选出版计划，同时出版纸质书。

图灵社区引进出版的外文图书，都将在立项后马上在社区公布。如果你有意翻译哪本图书，欢迎你来社区申请。只要你通过试译的考验，即可签约成为图灵的译者。当然，要想成功地完成一本书的翻译工作，是需要有坚强的毅力的。

最直接的读者交流平台

在图灵社区，你可以十分方便地写作文章、提交勘误、发表评论，以各种方式与作译者、编辑人员和其他读者进行交流互动。提交勘误还能够获赠社区银子。

你可以积极参与社区经常开展的访谈、审读、评选等多种活动，赢取积分和银子，积累个人声望。

ituring.com.cn